LA PRIMERA REVISTA DE PROGRAMACION EN CASTELLANO

SOLO PROGRAMADORES

ÑO III, NÚMERO 32.

975 PTS

Intranet con Linux

ternet

prendiendo Java

 nuevo de Borland

++ Builder

rogramación Windows

LE en isual Basic 4.0

ultimedia en la Web

hockWave

terfaces Gráficos

CL / TK erramientas abiertas

O. O.

omponentes en Delphi

Y además...

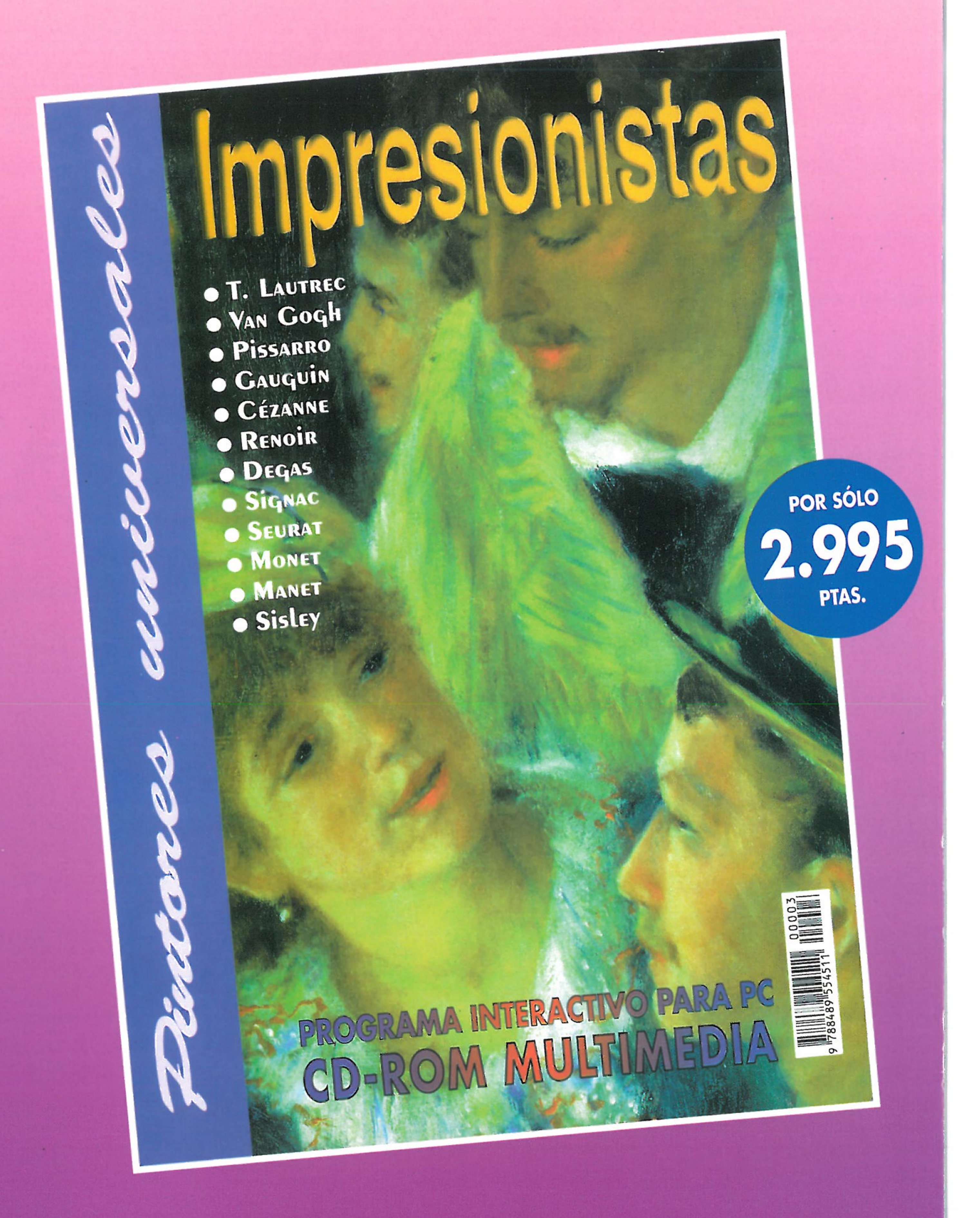
- Cliente / Servidor
- Generación de números aleatorios
- Novell Netware
- Universal Server de INFORMIX

CD ROM

- Índice en HTML de los títulos de la revista
- Slackware 96






Pintores universales
Impresionistas
T. Lautrec
Van Gogh
Pissarro
Gauguin
Cézanne
Renoir
Degas
Signac
Seurat
Monet
Manet
Sisley
POR SÓLO
2.995
PTAS.
PROGRAMA INTERACTIVO PARA PC
CD-ROM MULTIMEDIA
00003
9 788489 554511



Número 32
SÓLO PROGRAMADORES es una publicación de Tower Communications

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Directora Adjunta
Amelia Noguera
anoguera@towercom.es
Editora-Redactora
Francisca Guzmán
Colaboradores
Enrique Díaz, Miguel González Maestre, Arsenio Molinero, Carlos Álvaro, Enrique de la Lastra, Francisco José Abad Cerdá, Jose Antonio Collar, Javier Sarsa, Oscar Prieto Blanco, Ernesto Schmitz, Jose María Echevarría, Francisco García.

Jefe de Diseño
Fernando García Santamaría
fegarcia@towercom.es
Maquetación
Clara Francés
Tratamiento de Imagen
María Arce Giménez
Diseño de Portada
Fernando Núñez

Publicidad
Erika de la Riva (Madrid)
Tel.: (91) 661 42 11
Publicidad
Papín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Isabel Bojo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Filmación
Filma Dos S.L.
Impresión
G. Reunidas
Distribución
SGEL

Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
Director de Distribución
Enrique Cabezas García

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792

¿TE GUSTARÍA COLABORAR?

Desde que los ordenadores aparecieron y se consiguió que la electrónica con la que se construyen transmitiera nuestras órdenes a sus circuitos para obtener un resultado, muchas tecnologías han visto la luz y han invadido el mundo. La magnitud de los cambios que produjo la revolución industrial no tiene comparación con la revolución que están provocando las técnicas de adquisición, manipulación, almacenaje y tratamiento de la información. Quien posee la información posee la fuerza y ahora, quien no puede tratarla adecuadamente se pierde un trozo de la tarta.

En Sólo Programadores no queremos quedarnos atrás y avanzamos acorde a las tecnologías. Internet ha supuesto una inyección en la promoción de la investigación tecnológica, y nosotros, desde aquí, nos adherimos a la tendencia, al avance, y continuaremos tratando todos aquellos temas que vayan surgiendo en profundidad, con rigor informativo en contenidos y formas, y para ello contamos con fuentes inestimables.

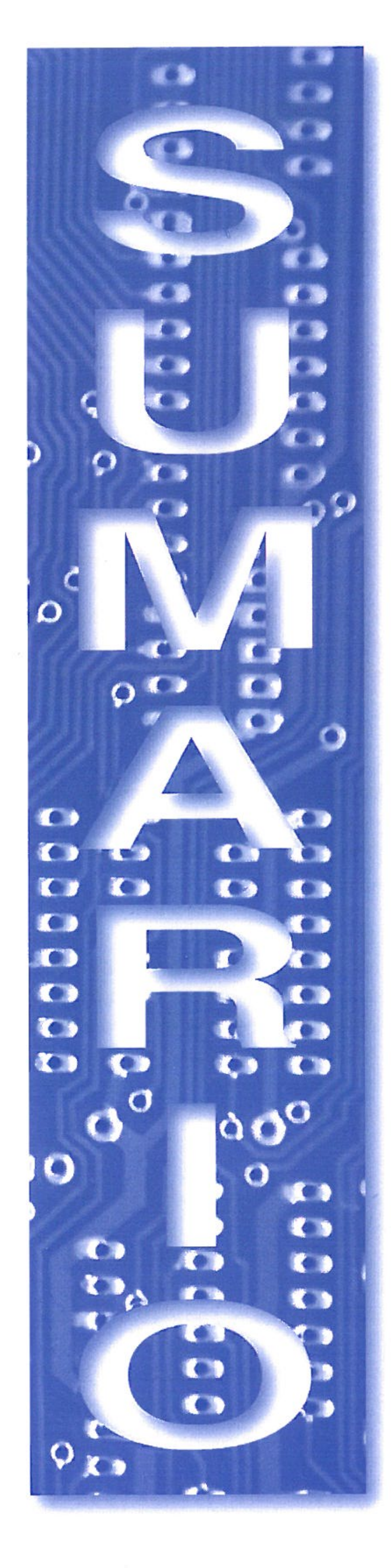
La universidad nos proporciona una, la empresa su complemento; en una se investiga, se innova, la otra se enfrenta día a día con la realidad del mercado y de la planificación de tiempos en el desarrollo de software y en el mantenimiento de las aplicaciones. Los lectores sois el tercer pilar.

Para vosotros, conseguiremos que el acceso a la información sea rápido y fácil; para vosotros, hemos creado un índice de todos los números de la revista que se actualizará y se mejorará cada mes; para vosotros, hemos conseguido reducir el precio mejorando la calidad. A lo largo de próximos números ireis encontrando cambios que nos gustaría que nos critiquéis o alabéis según os parezca, y para ello os instamos a que utilicéis la dirección de correo electrónico de Sólo Programadores, solop@towercom.es, donde podéis dirigir vuestras dudas, críticas, sugerencias y comentarios sobre la revista.

Una publicación es un ser vivo, que respira, se alimenta y se adapta al medio en que vive; en Sólo Programadores queremos que seais vosotros quienes decidáis la forma en que debemos seguir respirando. En este número hemos introducido temas nuevos, nuevos enfoques, y esperamos vuestras críticas para mantener, de entre todas las secciones, aquellas que más os gusten, y eliminar aquellas otras que no leais. Seguiremos atendiendo temas especializados de forma que resulten interesantes para todos, consiguiendo que desde nuestras páginas siempre se pueda obtener la información más completa, pero, para esto, necesitamos vuestra ayuda.

Sois nuestros colaboradores más importantes.

Con ello deseamos que Sólo Programadores siga constituyendo el mejor material de consulta en el trabajo y en los estudios. Y lo seguiremos logrando mes tras mes como hasta ahora. Mejorando, incluyendo temas según vuestros deseos, consiguiendo que los artículos sean escritos por expertos en cada una de las materias para, al final, mantener el listón tan alto como vosotros necesitáis para estar al día.


SUMARIO

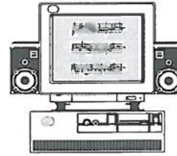
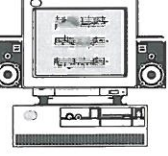

32


53

TECNOLOGÍA WEB:

SHOCKWAVE O LA INTERACTIVIDAD

Shockwave es una familia de plug-ins ampliamente consolidados entre los desarrolladores de Multimedia que permiten ejecutar en las páginas Web los documentos multimedia generados por los programas de la casa Macromedia.

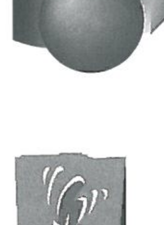

57

TECNOLOGÍA RELACIONAL-OBJETO:

UNIVERSAL SERVER DE INFORMIX

Las estructuras de datos aumentan su complejidad día a día; estudiamos la solución de Informix para un SGBD que introduce el paradigma de la orientación a objeto en el modelo de datos relacional.



65

INTERFACES GRÁFICAS:

HERRAMIENTAS ABIERTAS TCL/TK

En este artículo, tratamos las herramientas que componen Tcl/Tk que soportan distintas plataformas y se integran con otros lenguajes de programación como el lenguaje C.




68

CRIPTOGRAFÍA:

SI QUEREMOS QUE NUESTROS DATOS SEAN SECRETOS

La criptografía intenta conseguir seguridad y confidencialidad en las comunicaciones. En este artículo nos adentramos en el algoritmo de cifrado IDEA.




73

TECNOLOGÍA INTERNET:

CREAR UNA INTRANET EN LINUX

En este número comenzamos una serie de artículos que nos conducirán en el proceso de crear una intranet con Linux, proporcionando las pautas para llegar a aprender con la práctica todo sobre redes.




78

CORREO DEL LECTOR

CONSULTAS DE LOS LECTORES

Espacio dedicado a resolver los problemas surgidos a los lectores en diversos aspectos de la programación.





81

CONTENIDO DEL CD-ROM

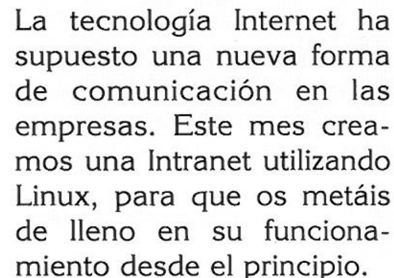
SLACKWARE 96, ÍNDICE DE LAS REVISTAS

El CD-Rom de este mes contiene la distribución Slackware 96, además de otras herramientas de programación y un útil índice con el que navegar por el contenido de todas las revistas y sus artículos.



REDES LOCALES: NETWARE DE NOVELL




La tecnología Internet ha supuesto una nueva forma de comunicación en las empresas. Este mes creamos una Intranet utilizando Linux, para que os metáis de lleno en su funcionamiento desde el principio.

Para que la revista sea aún mejor, os pedimos vuestra colaboración; mandadnos vuestros comentarios y sugerencias a la dirección de Sólo Programadores: solop@towercom.es
A partir del mes que viene os encontraréis con un nuevo diseño, más información y un índice con más contenidos, para que podáis buscar sin dificultad sobre el tema que os interese.

EL COFRE DE MICROSOFT

Microsoft anuncia su nueva generación de herramientas en el Día del Desarrollador

Microsoft ha presentado la edición empresarial de Visual Studio 97. Éstas y otras herramientas fueron anunciadas en el Día del Desarrollador, celebrado el pasado 20 de marzo.

Las nuevas versiones de las herramientas de desarrollo de Microsoft incluyen Visual Basic 5.0, Visual J++ 1.1, Visual InterDev 1.0, Visual C++ 5.0, Microsoft Transaction Server, Microsoft Visual FoxPro 5.0 y Office 97 Developer Edition.

Microsoft lanza estas versiones empresariales con el objetivo de proveer a los grupos de desarrolladores con herramientas para diseñar y construir soluciones multinivel (*multilier*). El fin primordial consiste en aprovechar las oportunidades de Internet y en su integración con las inversiones existentes en software y hardware.

Microsoft proporciona una sólida base para construir soluciones empresariales escalables basadas en aplicaciones de la familia BackOffice ejecutándose en Windows NT Server, declaró Jesús Pintado, jefe de producto de herramientas de desarrollo en Microsoft Ibérica.

Visual Basic 97 incorpora el soporte para grupos de desarrollo, diseño de soluciones a gran escala e integración con complejas bases de datos.

MICROSOFT VISUAL J++ 1.1

Visual J++ 1.1, la última versión de la herramienta de desarrollo para Java, representa otro de los diamantes de esta serie. Ésta incluye nuevas características que permiten a los desarrolladores integrar toda la potencia de Java con los sistemas existentes como las redes cliente/servidor y las bases de datos. Una edición de prueba de Visual J++ está disponible de forma gratuita en la dirección de Internet *http:// WWW.microsoft.com/ visualj/*. La edición profesional aparecerá como producto independiente o como parte de Visual Studio 97.

Visual J++ proporciona todas las herramientas necesarias para los desarrolladores en la creación de aplicaciones Java multiplataforma y applets. Estas herramientas cuentan con la capacidad de depurar applets dentro de un browser, el compilador Java más rápido del mercado, un editor personalizable y on-line.

EL DÍA DEL DESARROLLADOR

El pasado 20 de marzo, Microsoft celebró el Día del Desarrollador, jornada intensiva destinada a educar a los desarrolladores en las últimas versiones de herramientas de Microsoft para la integración del desarrollo cliente/servidor, Internet e intranet. Este evento tuvo lugar simultáneamente en más de 80 ciudades de 40 países (en España en Madrid y Barcelona).

Bill Gates, presidente de esta compañía, inauguró la sesión con una presentación vía satélite. Miles de desarrolladores se reunieron para compartir sus experiencias y conocimientos utilizando las últimas tecnologías y herramientas de Microsoft. Y es que en la actualidad, más de 5 millones de desarrolladores en todo el mundo utilizan las herramientas visuales de Microsoft para crear aplicaciones personalizadas y soluciones para empresas.

Los temas sobre los que giraron estas sesiones fueron el desarrollo del lenguaje multiplataforma, de componentes, de aplicaciones Web o la integración de sistemas de empresas y la creación de soluciones escalables. Según Microsoft, este año es el del Desarrollador.

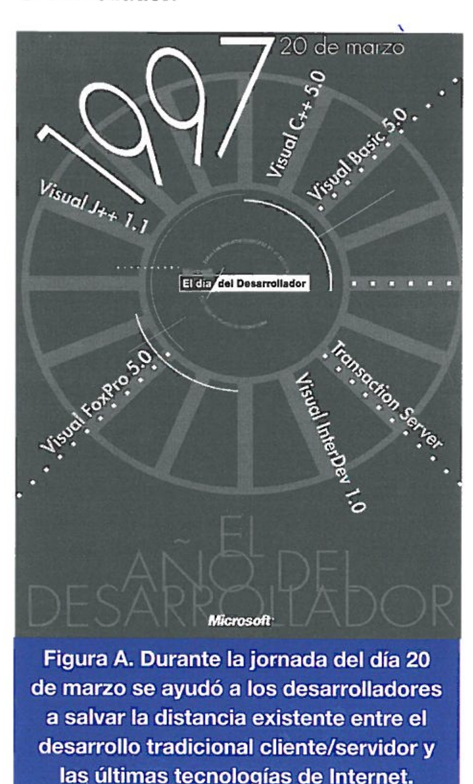


Figura A. Durante la jornada del día 20 de marzo se ayudó a los desarrolladores a salvar la distancia existente entre el desarrollo tradicional cliente/servidor y las últimas tecnologías de Internet.

LA ESTRELLA ES STAR: VIEW

Sterling Software lanza su nueva herramienta de acceso inteligente

Sterling Software ya ha sacado a la luz su nueva herramienta de acceso inteligente STAR: View, diseñada para operar vía Internet.

STAR: View permite el acceso remoto de usuarios desde cualquier lugar al *host* central de una organización, utilizando la infreaestructura de Internet. Ésta permite la conexión de un usuario remoto al *mainframe*, adecuándose a los protocolos TCP/IP y dando forma gráfica a entornos 3.270 en páginas HTML (*HyperText Markup Language*). Además, presenta interfaces Windows accesibles mediante cualquier navegador Internet convencional como Navigator, Explorer o Motif.

De este modo, se completa la familia STAR de la compañía (junto a STAR: Flaspoint y STAR: Flaspoint for theAS/400), la cual transforma automáticamente las pantallas tradicionales del *host* en paneles Windows fácilmente accesibles para los usuarios.

La familia de herramientas se halla orientada a evolucionar las aplicaciones con presentación 3270 o 5250 hacia un primer nivel de arquitectura cliente/servidor, transformando de manera automática las pantallas tradicionales de un *host* en paneles Windows. Así, se facilita la nevegación de pantallas, su contenido y su integración con el mundo del PC.

El deseo de Sterlin Software es ofrecer una respuesta eficaz y ergonómica a la apariencia clásica del entorno carácter, cuyo interfaz externo es poco manejable para usuarios acostumbrados a las ventajas del entorno Windows.

Para más información: (91) 590 35 05.

TRAS LA SEGUNDA WEB

Silicon Graphics busca un servicio más real y tridimensional

La compañía líder en gráficos 3D, Silicon Graphics, ha anunciado la creación de un software independiente de plataformas que acelera el desarrollo de una "Segunda Web" más interactiva y tridimensional. Por otra parte, la nueva versión del software de creación Cosmo Worlds VRML2.0 estará disponible para ordenadores personales a través de Windows NT.

El principal objetivo es proporcionar las tecnologías y productos necesarios para hacer posible una "Segunda Web" que atraerá a mucho más público que el actual y bidimensional, compuesto de texto e imágenes. Para ello, se ha liderado el desarrollo de un estándar sólido, escalable y abierto en forma de VRML 2.0 de mayor distribución para Windows 95 y Windows NT.

Tras diez años de experiencia, Silicon Graphics ha promovido el desarrollo de un estándar para la creación de un Web 3D interactivo -VRML. Este formato de archivo compacto es una tecnología innovadora con un gran ancho de banda. Combinado con el rápido crecimiento de las capacidades de visualización de gráficos 3D para el consumo, VRML proporcionará experiencias Web interactivas a un público mucho mayor.

Cosmo Worlds, el sistema profesional de creación para la Web VRML 2.0 potencia y mejora el proceso de trabajo completo. Estas sofisticadas y visuales herramientas de composición de escenas aceleran el ensamblaje de mundos VRML. La nueva versión de Cosmo Worlds para Silicon Graphics y Windows NT estará disponible a finales de 1997.

Para más información, es posible llamar al (91) 350 04 14.

LIBERACIÓN DE LA INFORMÁTICA DE REDES

Sun ofrece potentes aplicaciones en C/C++ y Fortran

Sun Microsystems, Inc. ha presentado recientemente la generación de Sun™ WorkShop™, el reconocido entorno de desarrollo de la compañía. Éste ofrece a los programadores de C/C++ y Fortran el mejor software empresarial para reducir el tiempo de desarrollo.

Adicionalmente, la compañía ha anunciado también el producto Sun™ WorkShop™ TameWare, versión 2.0, el principal sistema de gestión de código fuente para desarrollo en equipo.

Sun Visual WorkShop para C++ incluye un generador visual de interfaces gráficas de usuario (GUI) y Sun Performance WorkShop para Fortran permite a los desarrolladores ver y depurar complejas matrices de datos y algoritmos con WorkShop Data Visualizer.

Los potentes sistemas integrados de desarrollo empresarial WorkShop disparan la productividad y permiten a los equipos trabajar de forma más eficaz. Además, Sun Visual WorkShop para C++ y Sun Performance WorkShop para Fortran incluyen un MSRP el precio de 524.250 ptas. por puesto. Sun WorkShop TameWare tiene un precio de 194.250 ptas. por puesto.

Para obtener más información: *http://www.sun/workshop/*.

SUPERORDENADOR EN ESPAÑA
El CINEMAT adquiere el superordenador más potente del país

El software LAN Emulation de Olicom Inc. podrá ya utilizarse en los conmutadores Token-Ring de Cisco. Dichas compañías han anunciado la licencia de este producto en un acuerdo decisivo para el continuo desarrollo de red Token-Ring. Estos conmutadores, de altas prestaciones y segunda generación, incluirán la tecnología de *uplink* ATM.

LAN Emulation ha sido probado para que sea interoperativo con los *routers* y conmutadores de Cisco y soportar totalmente el protocolo SSRP (*Simple Server Redundancy Protocol*) para el *backup* de servicios LANE. Simultáneamente, Olicom participa en el programa ATM Associates de Cisco, lo que asegura que las tarjetas NICs ATM de Olicom son totalmente interoperativas con las redes Cisco.

La capacidad de conectar conmutadores Token-Ring vía *un uplink* al *backbone* ATM ha probado ser una necesidad crucial para los clientes de ambas empresas. Al integrar este nuevo software y las tecnologías IOS™, el Token-Ring se hallará disponible al mismo tiempo que el conmutador que se está desarrollando conjuntamente. El software LAN Emulation de Olicom fue elegido para el *uplink* ATM por su funcionalidad robusta y diseño portable.

Si desea obtener más información sobre los productos y servicios de Olicom y Cisco, podrá encontrarla desde Internet en las siguientes direcciones: *http://www.olicom.com* y *http://www.cisco.com*, respectivamente.

MUCHO MÁS QUE AS/400
J.D. Edwards e IBM amplían su acuerdo tecnológico

J.D. Edwards ofrece OneWorld, una aplicación que sobrepasa el entorno AS/400 y puede utilizarse indistintamente en otras plataformas de IBM.

J.D. Edwards es proveedor de soluciones informáticas para AS/400 de IBM desde hace 18 años. Ahora, con su nueva arquitectura Configurable Network Computing (CNC), permite que OneWorld sea operativo en cualquier plataforma, lo que capacita a los usuarios de este software a elegir el entorno de hardware que mejor satisfaga sus necesidades.

La sólida alianza estratégica entre IBM y J.D. Edwards consigue, así, soluciones tanto para los servidores de AS/400, (por los que se da soporte a más de 4.000 empresas), como para los servidores de los S/3000 y RS/600, así como a la familia de bases de datos DB/2.

El RISC Sistema/6000 de IBM aprovecha completamente la arquitectura CNC para proporcionar a sus clientes aplicaciones de software OneWorld en una amplia gama de entornos basados en red. En el entorno abierto proporcionado por el sistema operativo UNIX de IBM, los clientes pueden compartir datos y recursos con todas las plataformas que soporten a las aplicaciones OneWorld, con PCs y con otras estaciones de trabajo y servidores. De este modo, la inversión del cliente se verá protegida por la escalabilidad de la familia de productos RS/6000.

Por otro lado, los actuales usuarios del Sistema/3090, gracias a las continuas mejoras de este sistema, podrán aprovechar al máximo todas las variedades de las aplicaciones dinámicas y configurables de OneWorld.

LAS MEJORES SOLUCIONES PARA CAD/CAM/CAE
Silicon Works revoluciona EXPO´CAD 97

EXPO´CAD 97, la feria más importante de Europa en cuanto a productos CAD/CAM/CAE se refiere, ha sido conquistada por Silicon Graphics y sus nuevos ofertas para el mercado de la fabricación. Entre ellas destaca:

O2 MODELER, una potente línea de hardware de altas prestaciones y soluciones integradas de software, creada especialmente para este tipo de entornos. Incluye mapeado de texturas estándar y numerosas aplicacions que permiten la computación interactiva y colaborativa.

OCTANE es una nueva y potente familia de sistemas de sobremesa que proporciona gráficos de altas prestaciones, multiproceso sim´étrico (SMP) y un entorno de computación de 64-bits, además de una innovadora arquitectura de sistemas de altas prestaciones diseñada para satisfacer las futuras demandas.

Por último, la solución integrada para fabricantes SILICONWORKS combina potentes aplicaciones de terceros con sistemas informáticos visuales de la empresa y avanzadas herramientas de software. SiliconWorks moderniza y agiliza el desarrollo de productos, desde el concepto hasta la construcción de prototipos. Así, integra los sistemas avanzados de hardware de gráficos tridimensionales interactivos, medios digitales, visualización de alta gama y tecnologías de supercomputación con las aplicaciones de software de terceros más potentes de la industria en un entorno único, integrado y colaborador. Esta solución se complementa con más de 450 aplicaciones CAD/CAM/CAE.

EL COSTE DE PROPIEDAD AL MÍNIMO

Intel aporta conexiones a Red a 10/100 Mbps

Intel Corporation amplía su oferta de productos de red con la introducción de los adaptadores móviles EtherExpress PRO/100. Los nuevos adaptadores se dirigen al creciente segmento de mercado de los usuarios de ordenadores portátiles y reflejan el compromiso de Intel para aportar una capacidad de 10/100 Mbps a toda la Red, lo cual reduce el coste de propiedad.

Estos adaptadores móviles, que proporcionan una conectividad 10BASE-T y 100BASE-TX, caben en cualquier *slot* PCMCIA Type II y están disponibles en PC Card de 16 bits y CardBus de 32. Además, ofrecen altas prestaciones, una alternativa de bajo coste y fácil implementación para los empleados que utilizan una estación de *docking* o una combinación laptop/PC de sobremesa.

El coste total de propiedad también se reduce por la instalación *plug & play* y el soporte *HotSeap*. Una *suite* completa de *drivers* y un programa de instalación configura automáticamente el PC y el sistema operativo de red.

Los EtherExpress PRO/100 se lanzan al mercado en paquetes de una, cinco y 20 unidades. Su precio de venta al público es de 24.360 para la PC Card de 16 bits y 27.860 para la CardBus de 32 bits (20 unidades). De todos modos, hallará más datos sobre Intel desde el Web de la compañía en la dirección *http://www.intel.com/pressroom.*

LA SOLUCIÓN DE INFRESCO FRENTE AL SIGLO XXI

Nace OPAL 2.0, tecnología para integrar recursos de información empresarial

Infresco, subsidiaria de Computer Associates Internacional, presenta OPAL 2.0, una solución que impulsa a las empresas a adaptar sus sistemas de información a las tecnologías emergentes en Internet e Intranet. Las aplicaciones independientes y de visualizador de clientes de OPAL integran las fuentes de información actuales y futuras, liberando a los administradores de sistemas de las dependencias de *back-end* y plataforma de cliente.

OPAL 2.0 utiliza una aproximación crítica y de bajo riesgo que crea una intuitiva interfaz de usuario multimedia de negocios. Además, los productos OPAL incluyen clientes *run-time.*

OPAL Integrator, su entorno de diseño e integración, facilita el rápido desarrollo de nuevas aplicaciones. La utilidad Mapper configura fuentes de datos y crea páginas OPAL automáticamente. El servidor OPAL ofrece comunicación con *hosts* y bases de datos, eliminando la necesidad de suministrar esta conectividad en el cliente final. Los clientes y servidores se comunican a través de conexiones TCP/IP dedicadas, proporcionando enlaces de alta velocidad entre el cliente y las fuentes de información.

Por otra parte, se ha incorporado un grupo de tecnologías innovadoras: el administrador TM Snap-In, que garantiza una fácil expansión y DeltaUpdate TM, que simplifica la implementación.

De esta forma, Infresco se ha comprometido a proporcionar soluciones completas que introducen los sistemas existentes en el siglo XXI. Para Norm Worthington, presidente de la compañía, los sistemas de *hosts* y bases de datos de ayer y los servicios de objetos distribuidos de mañana están integrados en OPAL.

OPAL 2.0 estará disponible en el segundo trimestre de 1997. Sus clientes podrán operar como ejecutables independientes y en Netscape Navigator y MS Explorer bajo Windows 95/NT ó 3.X. OPAL Integrator opera con Windows 95/NT, mientras que el servidor lo hace con Windows NT o Unix.

Hallará más información sobre OPAL en la página Web de Infresco: *http://ww.infresco.com.*

NACE EL PRIMER AGENTE INTELIGENTE PARA INTERNET

Comet está dirigido a los usuarios del Nokia 9000

TeamWARE ha presentado el primer agente inteligente para Internet del Nokia 9000 Communicator. Su nombre código es "Comet" y permite a los usuarios del Nokia monitorizar nuevos mensajes de correo electrónico y los cambios en *web sites.*

Con ello, el suministrador líder en Europa de productos de *gruopware* (TeamWARE) ofrece funcionalidad móvil inteligente para correo electrónico, planificación personal, de grupos y recursos, así como notificaciones selectivas, utilizando los servicios de TeamWARE Office. Este conjunto de aplicaciones *groupware* funciona sobre plataformas servidor estándar y provee servicios de correo electrónico, agenda personal y corporativa, foros electrónicos y gestión documental.

Una vez activado, Comet actúa como un asistente personal que busca nueva información. Se ejecuta sobre el servidor de la red y realiza las solicitudes que el usuario seleccione al margen de los problemas que puedan ocurrir en la red e incluso cuando el usuario no esté conectado.

NOVEDADES

UNA APUESTA POR LA ALTA CALIDAD EN REDES DE VÍDEO

ALIANZA DE BAY NETWORKS Y FIRST VIRTUAL

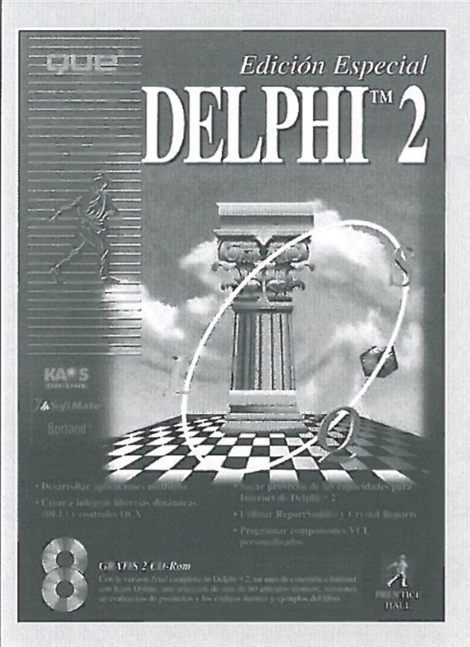
Bay Networks, Inc. y First Virtual Corporation han iniciado un acuerdo de distribución a nivel mundial para ofrecer toda la línea de productos de red de vídeo de First Virtual, dirigido al despliegue de aplicaciones multimedia de sobremesa en entornos ATM.

La ampliación de esta alianza permitirá a Bay Networks construir completas redes de vídeo para conectar las unidades existentes de conferencia de sala, así como participar en este sector, uno de los de mayor crecimiento.

La familia de First Virtual y la amplia gama de conmutadores de red de Bay Networks, como el Centillion 100 multi-LAN/ATM y el System 5000, permitirán a los clientes implementar redes de vídeo de gran calidad. Así, Bay ya podrá desplegar entornos completos de colaboración con vídeo de calidad profesional, conteniendo los tres elementos claves: conferencia, *caching* y *casting.*

La colaboración vídeo se está convirtiendo en una ventaja competitiva estratégica para muchas corporaciones y First Virtaul Corporation es líder en redes de vídeo de alta calidad. Si le interesa visitar su página Web, la dirección es *http://www.fvc.com.* También es posible consultar la dirección de Bay Networks: *http://www.baynetworks.com.*

LIBROS



Edición Especial Delphi TM 2

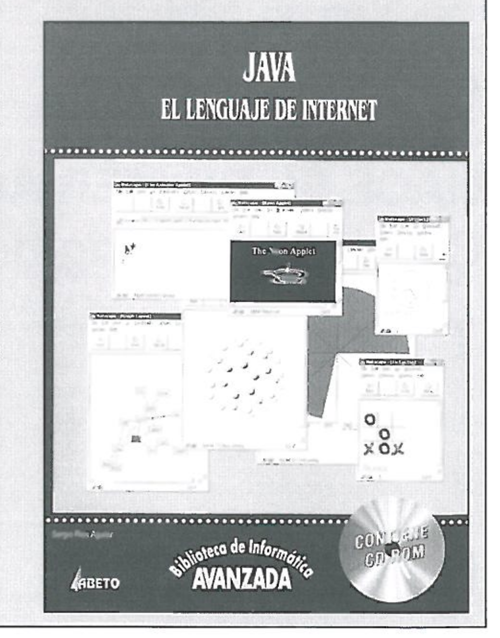
Delphi es un entorno de desarrollo que aprovecha las características del entorno gráfico de usuario de Microsoft Windows; como entorno de programación visual Delphi hace uso de los elementos que le proporcionan la potencia y versatilidad de los lenguajes visuales. Esta obra se adentra en estos elementos explicando la manera en que Delphi resuelve conceptos como objetos, programación controlada por sucesos y sus relaciones. Edición Especial Delphi TM 2 está dirigida a programadores y jefes de proyecto, lectores con un nivel intermedio-avanzado en programación y en metodologías de desarrollo que deseen ahondar en el conocimiento de las técnicas de programación visual.

Editorial: Prentice Hall
Autor: Jonathan Matcho y otros
680 páginas
Idioma: castellano
Nivel: intermedio-avanzado
Incluye CD-Rom

Java el lenguaje de Internet

Como guía de referencia rápida "Java el lenguaje de Internet" cumple todas las expectativas. Como manera sencilla y rápida de introducirse en el mundo de Java, esta obra supone un material de consulta estructurado donde la información tiene un modo de acceso fácil y cómodo.

El libro se divide en capítulos claramente diferenciados que tratan diversos aspectos de la programación con Java, desde las diferencias con C y C++, pasando por una guía básica del lenguaje, una referencia de características avanzadas que incluyen *threads*, tratamiento de excepciones, etc., hasta un estudio de la programación orientada a objetos, paradigma sobre el que Java se soporta, hasta llegar a los famosos *applets.*



Editorial: Abeto
Autor: Sergio Ríos Aguilar
192 páginas
Idioma: castellano
Nivel: principiantes
Incluye CD-Rom

VARIOS PROTOCOLOS EN NOVEL NETWARE

Enrique Díaz

Cuando un administrador planifica una red debe hacerlo como si ésta fuera a ser muy grande. Y esto es así aunque nuestra empresa sea más bien modesta y los recursos de que dispongamos sean, por decirlo de manera suave, escasos. Esto es precisamente lo que permite Novell 4.1. Podemos partir de una red meramente departamental y llegar a manejar una red mundial sin excesivos quebraderos de cabeza. Podríamos imprimir un documento en la impresora de aquí al lado o hacerlo en la impresora que nuestra empresa tiene en la planta 18 del Empire State.

¿Cómo se consigue esto? Básicamente y, consideraciones de *hardware* aparte, gracias al NDS y a la posibilidad de montar protocolos.

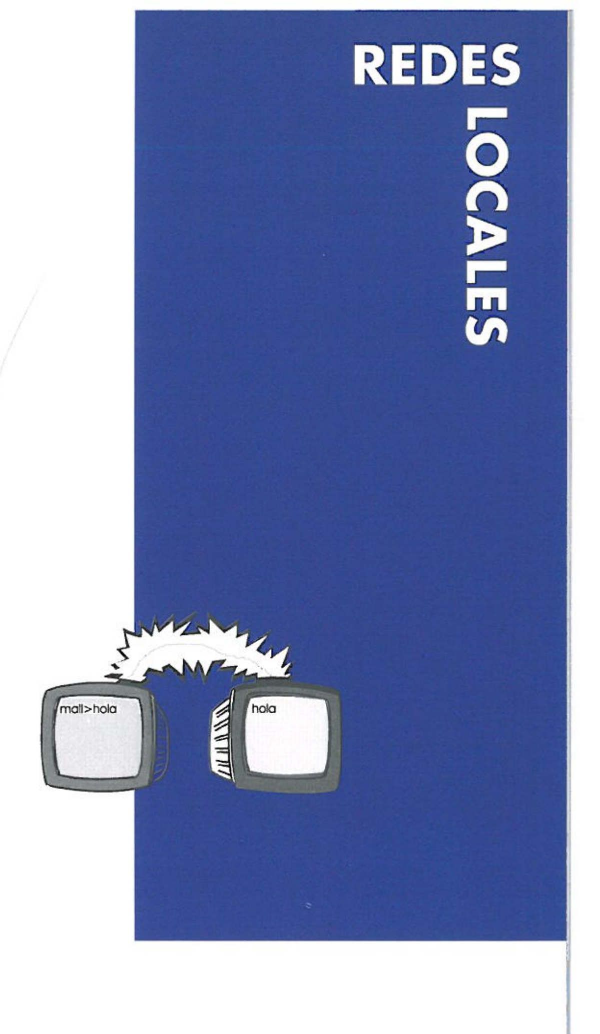
UN CASO PRÁCTICO

Supongamos que somos administradores de una red local. Nuestro ya veterano servidor Novell 3.11 está llegando al límite de sus capacidades y la empresa (¡al fin!) ha decidido proporcionarnos el dinero necesario para montar un servidor Novell 4.1.

En la instalación, una vez seleccionados los controladores de disco, habremos de seleccionar los controladores de la tarjeta de red. Cada tarjeta instalada en el servidor necesitará su controlador específico. Normalmente no tendremos problemas a la hora de encontrarlos ya que la propia Novell proporciona una gran variedad de controladores, aunque éstos no sean los œnicos ni tampoco las últimas versiones.

Así pues:

1) seleccionamos un controlador de la lista y pulsamos (Intro). En caso de que ninguno nos sirva podemos pulsar (Insert) y se nos solicitará una ruta alternativa donde buscar, normalmente en la unidad A:, pero es más recomendable haber copiado previamente a la instalación el controlador en algún directorio de C:. Tras seleccionar el controlador de cualquiera de estas maneras pulsamos de nuevo (Intro).
2) Antes de cargar efectivamente el controlador de nuestra tarjeta debemos asociarlo a los protocolos con los que vamos a trabajar. En la utilidad de instalación el protocolo IPX aparece ya seleccionado y veremos dos ventanas, una con los protocolos que podemos seleccionar en este momento, que son Appletalk y TCP/IP. En la otra ventana veremos los parámetros del protocolo IPX: IRQ, puerto, DMA si lo tiene, etc.
3) Debemos comprobar que estos parámetros se corresponden con la configuración de nuestra tarjeta, y si no es así modificarlos para que el controlador se cargue correctamente. Dicho en otras palabras, cuando se instala físicamente la tarjeta de red y se configura, hay que apuntarse el puerto, IRQ, etc. en un Post-It y pegarlo en el monitor. Esto es válido tanto para la tarjeta como para cualquier dispositivo (CD-ROM, unidad de cinta, etc.) que queramos utilizar.
4) Después de dar los parámetros correctos de la tarjeta pulsamos (Intro) para continuar. En este punto el servidor nos preguntará si deseamos seleccionar algún controlador de red adicional, a lo que responderemos sí o no en función de si tenemos alguna otra tarjeta de red.



Los servidores Novell son muy flexibles, uno se puede conectar a ellos desde casi cualquier plataforma, desde un terminal tonto hasta OS/2 o Windows 95, pasando por estaciones sin disco duro ni disquetera. Son servidores pensados para 'ver' y 'dejarse ver' muy fácilmente.

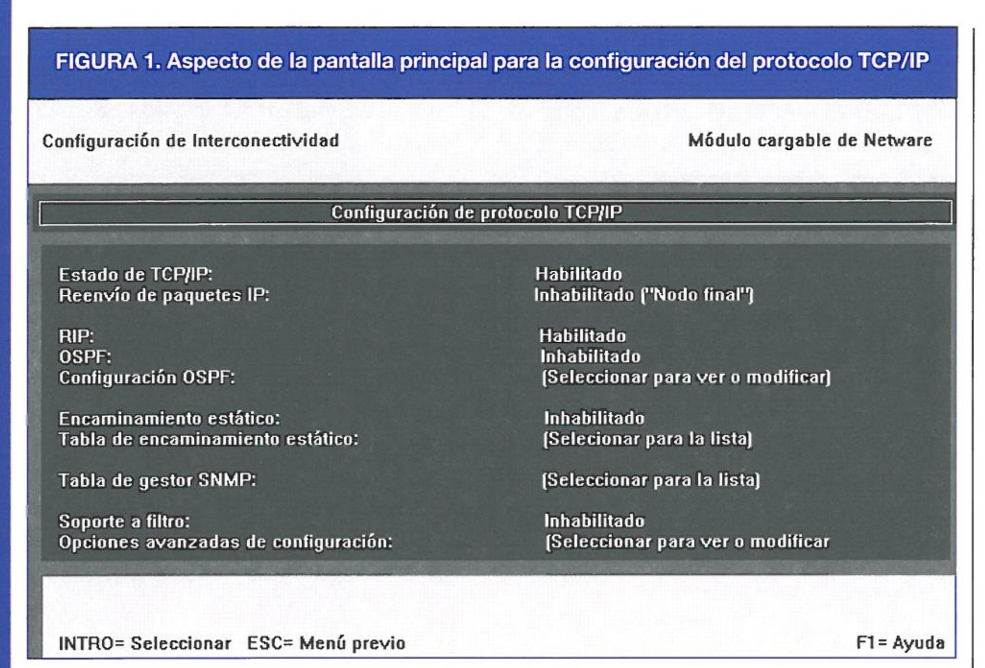


FIGURA 1. Aspecto de la pantalla principal para la configuración del protocolo TCP/IP

5) A continuación se asocia el protocolo a la tarjeta y se muestran una serie de pantallas donde elegiremos los tipos de tramas con los que trabajará el servidor. Si todavía no queremos hacer esto, pulsaremos (F3) y los controladores se visualizarán antes de ser cargados.

Vamos a suponer que hemos montado una trama Ethernet 802.2, que hemos llamado a la tarjeta NE802.2 y le hemos dado el número de red (en hexadecimal) 00FABADA. Terminamos de montar el servidor, lo levantamos y... no nos 've' nadie, ni nosotros 'vemos' al servidor 3.11. ¿Qué ha pasado?

Muy sencillo, el servidor 3.11 maneja tramas Ethernet 802.3 y nosotros para poder manejar el NDS de la 4.1 necesitamos tener obligatoriamente la trama 802.2. En cuanto a los clientes, no nos 'ven' porque sólo están configurados para conectarse al servidor 3.11.

¿Qué haremos? ¿Montar una segunda tarjeta en el servidor y en todos los clientes para que se 'vean' todos a todos? Pues eso es exactamente lo que vamos a hacer, con la salvedad que las tarjetas no serán físicas, sino lógicas. Y aquí es donde le encontramos toda la gracia al ODI (*Open Datalink Interface*) que implementa Netware.

ODI permite que los diferentes protocolos de red se combinen en diferentes encapsulaciones o tramas. Esto es, que podemos tener varias ÔredesÕ que viajan por el mismo cable. Más adelante veremos cómo funciona esto. De momento vamos a llevarlo a la práctica.

Cuando montamos la primera tarjeta, en realidad no montamos una tarjeta física, lo que hicimos fue montar la primera tarjeta lógica sobre una tarjeta física. Ahora lo que haremos es montar una segunda tarjeta lógica con la trama Ethernet 802.3, y para ello no utilizaremos ninguna utilidad; lo haremos directamente desde la línea de comandos de la consola del servidor. La forma general para hacer esto es:

Load controlador_de_tarjeta port=número_de_puerto frame=frame_elegido name=el_nombre_que_queramos.

Lo que aplicado a nuestro ejemplo (y aquí es donde echamos mano de nuestro Post-It) sería:

Load ne2000 port=300 int=3 frame=ethernet_802.3 name=ne802.3.

Ahora sólo hay que enlazar la tarjeta con el protocolo. Su fórmula general es:

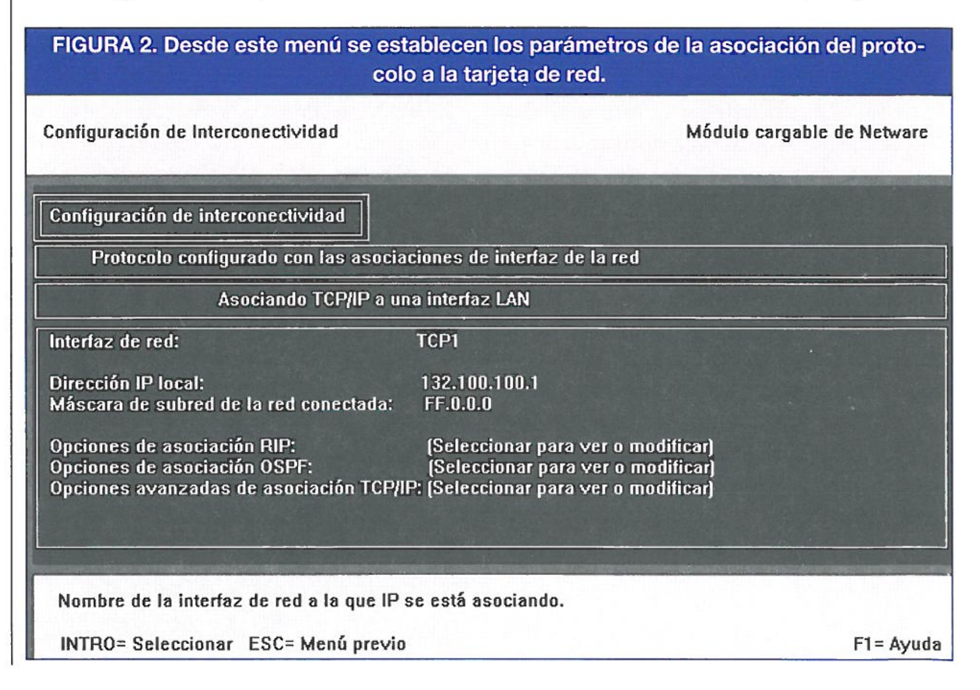
bind protocolo to controlador_de_tarjeta net=número_de_red

y en nuestro ejemplo:
bind ipx to ne2000 net=CAFÉ

El servidor nos mostrará numeradas las tarjetas lógicas que tenemos montadas y nos preguntará a cuál queremos enlazar el protocolo. En nuestro caso es la número 2.

En este momento podríamos 'ver' al servidor 3.11 y los clientes podrán 'vernos' a nosotros, aunque sería conveniente configurar los clientes para soportar tramas 802.2 y aprovechar de esta manera las ventajas que ofre-

FIGURA 2. Desde este menú se establecen los parámetros de la asociación del protocolo a la tarjeta de red.



ce esta trama sobre la 802.3 y también para poder manejar el NDS de Netware.

Veamos cómo funciona lo que hemos hecho. Cuando el servidor recibe un paquete, éste se analiza y se envía a la tarjeta lógica adecuada en función del protocolo y la trama que tenga. La tarjeta desensamblará y desencriptará el paquete de forma que el servidor pueda entender la petición que se le hace. La respuesta se procesa y se envía por el protocolo, trama y tarjeta adecuados.

Si, por ejemplo, tuviéramos una tarjeta de red adicional que conectara el servidor con un segundo segmento de red con topología *Token Ring,* el servidor podría recibir una petición enviada por la trama 802.2 y dirigida al servidor del segmento *Token Ring,* cambiarla a la tarjeta para *Token Ring* y darle el encapsulado adecuado, esto es, la trama 802.5. Y todo sin que nosotros tengamos que hacer nada. Tan sólo montar los protocolos y tramas adecuadas.

De hecho, al montar dos tarjetas físicas y sus correspondientes lógicas, agregamos a nuestro servidor la función de *router* y también sin hacer nada. Más fácil imposible.

ODI nos permite tener varias redes viajando por un solo cable

MONTAR UN PROTOCOLO TCP/IP

Hasta ahora lo único que hemos hecho ha sido montar un solo protocolo (IPX) con diferentes encapsulaciones o tramas. Siguiendo con nuestro ejemplo anterior, supongamos ahora que nuestra red sigue creciendo y que debemos preparar nuestro servidor 4.1 para atender peticiones de un servidor UNIX, que está en otro segmento de red, vía TCP/IP.

En esta ocasión vamos a utilizar la utilidad *Inetcfg.*

En Netware 4, TCP/IP se construye con una serie de archivos NLM (Módulos cargables de Netware) ubicados en el directorio SYSTEM del volumen SYS, y archivos de base de datos que se instalan en el directorio ETC\SAMPLES, del mismo volumen. Los archivos NLM necesarios son estos:

- TCPIP.NLM, archivo TCP/IP.
- SNMP.NLM, archivo de protocolo simple para la gestión de red.
- SNMPLOG.NLM, archivo de consola TCP/IP.
- IPCONFIG.NLM, archivo para configurar el encaminamiento IP.

Los archivos de base de datos habrán de ser configurados y TCP/IP los utilizará para la conversión de datos internos, de manera que resulte más sencillo trabajar con ellos.

Así pues, para construir un subsistema TCP/IP debemos:

1. Configurar el protocolo y asociarlo a una tarjeta de red.
2. Configurar los archivos de base de datos.
3. Cargar en el servidor el módulo TCPIP.NLM.

Novell 4.1 no tiene software de cliente para TCP/IP pero nos servirá cualquiera del mercado y los hay incluso gratis.

INETCFG es una utilidad de configuración de interconectividad que sirve para configurar las tarjetas y enlazarlas a protocolos sin tener que hacerlo manualmente desde la línea de comandos de la consola del servidor. Desde aquí podremos modificar configuraciones existentes y crear otras nuevas.

Para ello:

- Tecleamos LOAD INETCFG en la consola del servidor. Aparecerá el menú Configuración de interconectividad.
- Seleccionamos la opción Protocolos y pulsamos (Intro), con lo que aparece el menœ Configuración de protocolo, donde seleccionamos TCP/IP y pulsamos (Intro). Se nos presentará el menú Configuración de protocolo TCP/IP.
- En la Opción Estado de TCP/IP, pulsamos (Intro), seleccionamos Habilitada y pulsamos (Esc) para activar la selección. Si el servidor va a ser también un router debemos habilitar también la opción Reenvío de paquetes IP. No es necesario cambiar el resto de los parámetros.

FIGURA 3. Información que mostrará el comando config tras montar el protocolo IPX en diferentes encapsulaciones.

```
BIBI:config
File server name: BIBI
IPX internal network number: 0000DED0

Novell NE2000
   Version 3.25      June 17, 1993
       Hardware setting: I/O Port 300h to 31Fh, Interrupt Ah
       Node address: 080000174854
       Frame type: ETHERNET_802.2
       Board name: NE802.2
       LAN protocol: IPX network 00FABADA

Novell NE2000
   Version 3.25
       Hardware setting: I/O Port 300h to 31Fh, Interrupt Ah
       Node address: 080000174854
       Frame type: ETHERNET_802.3
       Board name: NE802.3
       LAN protocol: IPX network 0000CAFE
BIBI:_
```

FIGURA 4. La pantalla de seguimiento del Router nos muestra los paquetes que se envían y reciben a través de las diferentes tramas.

```
OUT [00FABADA:FFFFFFFFFFFF]  7:02:59pm  00000001  1/2
OUT [00000001:FFFFFFFFFFFF]  7:02:59pm  00FABADA  1/2
IN  [00FABADA:000000000001]  7:03:01pm  BIBI            1
OUT [00FABADA:FFFFFFFFFFFF]  7:03:29pm  BIBI            1  BIBI       2
OUT [00000001:FFFFFFFFFFFF]  7:03:29pm  BIBI            1  BIBI       2
IN  [00FABADA:000000000001]  7:03:29pm  BIBI            1
IN  [00FABADA:000000000001]  7:03:56pm  BIBI            1
OUT [00FABADA:FFFFFFFFFFFF]  7:03:58pm  00000001  1/2
OUT [00000001:FFFFFFFFFFFF]  7:03:58pm  00FABADA  1/2
<Use ALT-ESC or CTRL-ESC to switch screens, or any other key to pause>
```

- Volvemos al menú principal guardando los cambios y seleccionamos Asociaciones para agregar el protocolo TCP/IP a la tarjeta de red. Aparecerá una ventana con la lista de asociaciones configuradas entre protocolo y controlador de tarjeta.

- Creamos una nueva asociación pulsando (Insert) y seleccionando TCP/IP. A continuación seleccionar el controlador de tarjeta sobre el que montaremos el protocolo. Nos aparecerá una nueva ventana para configurar los parámetros del protocolo TCP/IP.

- En Dirección IP local introducir la dirección IP asignada a la tarjeta. Esta dirección debe ser única en la interred y su formato es cuatro números, decimales o hexadecimales, separados por puntos (por ejemplo 132.100.100.1).

- En Máscara de subred introducimos un número en decimal o hexadecimal que asocia máscara y tarjeta. Este valor debe ser el mismo para todos los nodos que se conecten a la red vía TCP/IP.

- El resto de los parámetros no se modifica y salimos de Inetcfg guardando los cambios.

CONFIGURAR ARCHIVOS DE BASE DE DATOS

Para la conversión de datos internos se utilizan los archivos HOSTS, NETWORKS, PROTOCOL y SERVICES. Su utilización no es estrictamente necesaria, y si se usan ocuparán algo de memoria, pero resultan útiles para traducir a nombres sencillos datos como direcciones IP.

Una manera rápida y fácil de configurar estos archivos es copiar unos ejemplos de estos archivos del directorio ETC\SAMPLES al directorio ETC y modificarlos con cualquier editor de texto. Y para ser más rápidos todavía, no será necesario modificar los archivos PROTOCOL y SERVICES cuando se copian al directorio ETC.

A continuación daremos una breve descripción de estos archivos y su formato general.

HOSTS

Establece la correspondencia entre nombre de host y dirección IP. Su formato general es:

Dirección_ip Nombre_del_host. [Alias].

Dirección_ip es la mencionada dirección de 4 bytes separados entre sí por un punto.

Nombre_del_host es el nombre que asociamos con la dirección indicada.

Alias es un parámetro opcional y consiste en un nombre alternativo.

NETWORKS

Relaciona nombres de red con la dirección de la red. Su formato general es:

Nombre_de_red Número_de_la_red [Máscara_de_la_red] [Alias].

Nombre_de_red es el nombre que asociaremos al Número_de_la_red, debe ser único y no puede tener números ni espacios.

Número_de_la_red es el número asociado al nombre de la red, por ejemplo 132.100.0.0.

Máscara_de_la_red es opcional y corresponde al número de subred de la red.

PROTOCOL

Especifica los protocolos usados en la interred. Su formato general es:

Nombre_del_protocolo Nœmero_del_protocolo [Alias].

Nombre_del_protocolo es un nombre de protocolo Internet asociado a su número. Este nombre tampoco puede tener números ni espacios.

Número_del_protocolo es el número de protocolo Internet asociado al nombre.

SERVICES

Relaciona nombres de protocolos con nombres de servicios tales como FTP, TELNET... Su formato general es:

Nombre_del_servicio Número_de_puerto/Nombre_del_protocolo [Alias].

Nombre_del_servicio es el nombre asociado al número del puerto y protocolo indicados. Este nombre no puede tener espacios ni números.

Número_de_puerto es el número de puerto Internet que utiliza el servicio.

Nombre_del_protocolo es el protocolo al que está asociado el servicio.

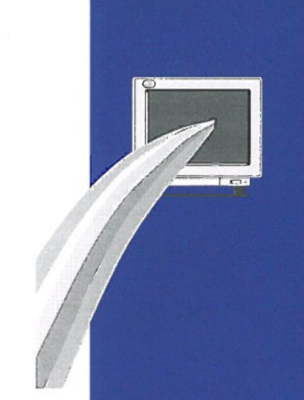
Para finalizar, solo nos queda teclear en la consola del servidor

LOAD TCPIP

para activar el soporte para TCP/IP.

PRIMEROS PASOS

Oscar Prieto



Java es un lenguaje con tintes revolucionarios, creado por James Gosling e introducido por Sun Mycrosystems en junio de 1995. Fue diseñado como un lenguaje de sintaxis muy similar a la del C++, pero con la idea de que fuera pequeño, simple y portable a través de diferentes plataformas y sistemas operativos (tanto a nivel de código fuente como nivel binario), para lo cual se implementó como un lenguaje híbrido, a medio camino entre compilado e interpretado. A grandes rasgos, se compila el código fuente *.java a un formato binario *.class (*bytecode*), que luego es interpretado por una máquina virtual java, la cual debe estar implementada para la plataforma particular usada. Para obtener más información acerca del nacimiento y características generales del lenguaje podéis consultar la referencia [Echeva-1].

En Java podemos construir dos tipos básicos de programas: utilizarlo como un lenguaje de propósito general para construir aplicaciones independientes o emplearlo para crear *applets*. La pregunta lógica que os vendrá a la mente será acerca del significado de la palabreja *applet*. Un *applet* se puede definir como una pequeña aplicación accesible en un servidor Internet, que se transporta por la red, se instala automáticamente y se ejecuta in situ como parte de un documento web. En esta serie, en primer lugar nos dedicaremos a consolidar el conocimiento del lenguaje, antes de dedicarnos a la programación de applets. En [Echeva-2] podemos encontrar referencias acerca de cómo incrustar un applet en un documento HTML.

Puesto que Java es un lenguaje orientado a objetos antes de meternos de lleno en él vamos a repasar algunos aspectos básicos de esta filosofía de programación. Paralelamente, observaremos cómo se diseñaron en java un lenguaje que tenía como objetivo ser simple y de fácil aprendizaje. Al mismo tiempo nos iremos introduciendo, brevemente, en la sintaxis del lenguaje.

JAVA COMO LENGUAJE ORIENTADO A OBJETOS.

Superficialmente, el término orientado a objetos significa que el *software* se organiza como una colección de objetos discretos que contienen tanto estructuras de datos como métodos que actúan sobre ellas, en oposición con la programación procedimental, donde la relación entre los datos y las funciones que los manejan es muchísimo más débil.

Entre las características principales que debe soportar un lenguaje para que sea orientado a objetos se suelen citar: encapsulación, herencia y polimorfismo.

ENCAPSULACIÓN

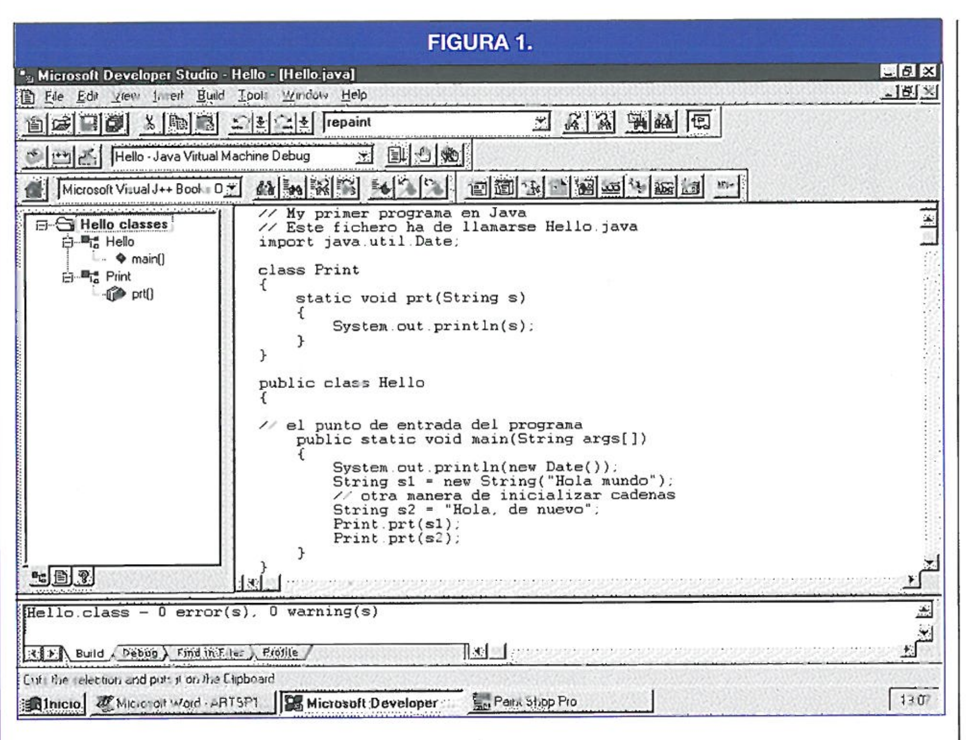
La encapsulación consiste en separar los aspectos externos del objeto, a los cuales pueden acceder otros objetos, de los detalles de implementación del mismo, que quedan ocultos para los demás. En Java la unidad primigenia de encapsulado es la clase, cuya sintaxis es la siguiente: class NombreClase{...} Veamos un ejemplo:

```
class Bombilla{
// Atributos
//////////////////////////////
// Variable de clase, es
// compartida por todos los obj.
```



Durante el proceso de instalación de Windows NT 4.0 nos encontramos con la frase: "... la tecnología JAVA puede llegar a producir la muerte ...", en esta serie que ahora comienza nos introduciremos en este "peligroso" mundo.

FIGURA 1.




```
static int valorPts = 100;
// Variable de instacia, cada obj.
// tiene su propia copia
int potencia = 60;
// Comportamiento
//////////////////////////////
// Métodos de instancia, operan
// sobre un objeto en particular.
void Encender(){/*...*/}
void Apagar(){/*...*/}
// Método de clase, opera sobre
// toda la clase, no es necesario
// declarar un objeto para usarlo
static int Precio
{return valorPts; }
}

// Creación de un objeto
// la palabra clave es new.
Bombilla b = new Bombilla();
// Acceso a un método del objeto
b.Apagar();
// Acceso a un método de clase
// Primera manera:
int pts = b.Precio();
// Forma más utilizada:
int pts = Bombilla.Precio()
```

Por simplicidad, se decidió que la declaración y la definición de la clase vinieran juntas, no como en C++ que cuenta con los archivos de cabecera y los archivos de definición. Asimismo, en Java se eliminaron las directivas del preprocesador, así como el uso de variables y funciones globales (que tantos problemas dan en el mantenimiento de grandes programas). El nivel siguiente de encapsulado es el de fichero, donde sólo una de las clases allí definidas será visible al usuario de ellas (en realidad, no es tan simple, pero ya lo estudiaremos posteriormente). El nivel más alto de encapsulación viene dado por la utilización de *packages* que son, simplemente, una colección de ficheros. Por tanto un *package* se puede entender como una colección de clases destinada para su uso por terceros, de manera análoga a como otros lenguajes utilizan las librerias.

HERENCIA

Mediante la herencia se puede definir una clase que después se irá refinando sucesivamente para producir subclases. Todas las subclases poseen, o heredan, todas y cada una de las propiedades de su superclase y añaden, además, sus propiedades exclusivas. Todas las clases usadas en Java derivan en última instancia de una clase base llamada Object. Por otro lado, se decidió eliminar la posibilidad de herencia múltiple, en la que una clase puede heredar de varias superclases al mismo tiempo, a la vista de la complejidad léxica que añadía (como se había comprobado en C++). Para paliar las limitaciones de este enfoque, se utilizan interfaces (que pueden ser usados simultáneamente con la herencia y en número arbitrario) que, en pocas palabras, son un conjunto de métodos que deberá definir la clase que implemente el interface. Sintaxis:

```
// Herencia
// class derivada extends base{...}
// Uso de un interface
// class x implements y{...}
```

POLIMORFISMO

Significa, por un lado, que un mismo método puede comportarse de manera distinta en clases distintas; y por otro, que una misma clase puede tener varios métodos con el mismo nombre, pero que varíen en su lista de argumentos. Java lo soporta, pero eliminó la sobrecarga de operadores presente en C++, por considerar que producía código dificil de leer y mantener. La única excepción a lo dicho se produce con el operador + que utilizado con objetos String, realiza su concatenación.

HANDLES VERSUS TIPOS BÁSICOS

En Java existen 8 tipos primitivos de datos, que no son objetos, aunque el lenguaje cuenta con uno equivalente para cada uno de ellos, ver tabla 1.

Como vemos en la tabla, el tamaño de estos tipos está fijado, siendo independiente del microprocesador y del sistema operativo sobre el que esté implementado. Esta característica es esencial para el requisito de la portabilidad entre distintas plataformas.

¿Por qué existen estos tipos primitivos y no sólo sus objetos equivalentes? La razón es sencilla, por eficiencia. Estos tipos básicos son almacenados en una parte de la memoria conocida como el *Stack*, que es manejada directamente por el procesador a través de un registro apuntador (*stack pointer*). Esta zona de memoria es de rápido acceso, pero tiene la desventaja de que el compilador de java debe conocer, cuando está creando el programa, el tamaño y el tiempo de vida de todos los datos allí almacenados para poder generar código que mueva el *stack pointer*, lo cual limita la flexibilidad de los programas. En cambio, los objetos son creados en otra zona de memoria conocida como *Heap*. Esta zona es de propósito general y, a diferencia del *Stack*, el compilador no necesita conocer ni el tamaño, ni el tiempo de vida de los datos allí alojados. Este

TABLA 1.

Tipo básico	Tamaño	Mínimo	Máximo	Obj.Equivalente
boolean	1 bit	vacío	vacío	Boolean
char	16 bits	Unic. 0	Unic. 2^{15}	Character
byte	8 bits	-128	+127	Byte
short	16 bits	-2^{15}	$2^{15} - 1$	Short
int	32 bits	-2^{31}	$2^{31} - 1$	Integer
long	64 bits	-2^{63}	$2^{63} - 1$	Long
float	32 bits	IEEE754	IEEE754	Float
double	64 bits	IEEE754	IEEE754	Double

enfoque es mucho más flexible pero, en contraposición, el tiempo de acceso a esta zona es más elevado que el necesario para acceder al *stack*.

Aunque en la literatura se comenta que Java eliminó los punteros, esta afirmación es inexacta. Lo que no se permite es la aritmética de punteros. Cuando estamos manejando un objeto en Java, realmente estamos utilizando un *handle* a dicho objeto. Podemos definir un *handle* como una variable que contiene la dirección de memoria donde se encuentra el objeto almacenado. Ahora ya vemos más clara la sintaxis utilizada para crear un objeto:

```
Bombilla bulb; // Creo el handle
// Creo el objeto en el heap
// y asigno su dirección al handle
bulb = new Bombilla();
```

Utilizando una analogía, podemos ver un handle como el mando a distancia de una televisión y al objeto como la televisión. Si tenemos el mando, pero no la televisión, no podemos encenderla, apagarla, etc. (es decir, ejecutar los métodos del objeto). Si tenemos el objeto, pero ningún handle ligado a él, un misterioso recolector de basura (garbage collector), implementado en el lenguaje, automáticamente eliminará de memoria al objeto. Este recolector representa una de las características más importantes del lenguaje: la gestión automática de memoria; esto es, se acabó el liberar la memoria manualmente como, por ejemplo, en C++ donde utilizamos el operador delete. En Java sólo nos preocupamos de crear los objetos.

ALCANCE DE LOS TIPOS DE DATOS

El alcance determina tanto la visibilidad como el tiempo de vida de los nombres definidos. En C, C++ y Java, el alcance es determinado por el uso de llaves {/*...*/}. Por ejemplo:

```
{
        int x = 14;
        //sólo x accesible
        {
                int y = x = 13;
                // X e y accesibles
        }
        // y fuera de alcance
        // sólo x accesible
}
```

Los objetos java no tienen el mismo tiempo de vida que los tipos primitivos. Su duración viene determinada por los handles a ellos ligados. Estudiemos el siguiente ejemplo:

```
{
Bombilla b1;
        // sólo b1
accesible
{
        Bombilla b2;
        b2 = new
Bombilla();
        // b1 y b2
accesibles
        // obj. en el
heap
        b1 = b2;
        // b1 contie-
ne la misma
        // dirección
de memoria que b2
        }
        // b2 fuera
de alcance
        // b1 todavía
accesible
        // el objeto
sigue en el heap.
}
```

Después del final de esta última llave tanto los handles b1 como b2 se encuentran fuera de alcance, pero el objeto sigue en el heap. Esta memoria será liberada en la siguiente pasada del recolector de basura, que está funcionando paralelamente a la ejecución del programa con un nivel bajo de prioridad. Este recolector examina todos los objetos que han sido creados mediante new y comprueba si tienen ligados algún handle, en caso negativo, liberará la memoria a ellos asignada para que pueda ser usada por otros objetos.

EL PRIMER PROGRAMA JAVA

Con los conceptos hasta ahora vistos ya podemos escribir nuestro primer programa en Java. Como herramienta de trabajo vamos a utilizar el JDK 1.02 de Sun MycroSystems, que fue entrega-

do a los lectores de esta revista en el número 23.

Como primer paso crearemos un fichero de texto ASCII, por ejemplo con el editor del DOS, al que llamaremos Hello.java. Un fichero de código fuente a menudo es llamado una unidad de compilación. En cada unidad de compilación puede existir como mucho una clase con el especificador de acceso public, el cual la hace visible al exterior del package (en próximos artículos comentaremos en profundidad este concepto). El resto de las clases permanecerán ocultas al exterior del package y su único objetivo será el de ayudar a la clase pública a realizar su cometido. El fichero de código fuente ha de ser llamado con el nombre de la clase pública (respetando las mayúsculas), seguido de la extensión java. Para compilar el fichero Hello.java, teclearemos desde la línea de comandos: javac Hello.java

Si no hay errores obtendremos un fichero de extensión .class por cada clase definida en la unidad de compilación. En este caso obtenemos: Print.class y Hello.class. Para ejecutar el intérprete, tecleamos sin más: java Hello y obtendremos la salida por pantalla del programa.

Vamos a estudiar detalladamente este ejemplo para ver qué conceptos nuevos han aparecido. La línea import java.util.Date; avisa al compilador de que quiero utilizar la clase standard Date en mi programa. El compilador por defecto importa el package java.lang, el cual contiene clases útiles de uso general, como la clase String, usada en este ejemplo. Siguiendo con el ejemplo, vemos definida una clase Print, la cual sólo contiene una función estática llamada prt, la cual no retorna ningún valor y que acepta como argumentos una cadena, que será impresa por el dispositivo de salida estándar (la pantalla). Para realizar esta tarea utilizamos un objeto estático out perteneciente a la clase System (incluida en java.lang), sobre el que ejecutamos el método println().

Java elimina la utilización de funciones y variables globales

Siguiendo con el ejemplo nos encontramos con la definición de la clase Hello, que declara y define un único método. Cuando utilizamos Java para crear una aplicación de propósito general, una de las clases de la unidad de compilación debe llamarse como el fichero y además dicha clase debe tener definida una función de la forma:

```
public static void main(String args[])
```

Esta función es el punto de entrada del programa y nos está dando a entender que esta clase puede ser usada como módulo inicial en un programa. El argumento de este método es un array de cadenas que nos sirve para poder pasar parámetros al programa mediante la línea de comandos de forma bastante similar al C. Esta función es estática dado que va a ser llamada inicialmente al comenzar el programa, no existiendo en ese punto todavía ningún objeto creado. Esta función es pública porque tiene que poder ser accedida desde fuera del fichero. En la primera línea del cuerpo del método main nos encontramos con la sentencia System.out.println(new Date()); donde encontramos un par de ideas importantes:

El garbage collector realiza la gestión automática de la memoria, evitando la utilización de operadores como detele, de C++

- Aparece el concepto de sobrecarga de funciones que, básicamente, significa el hecho de que dos métodos tengan el mismo nombre, pero se diferencien en su lista de argumentos. En este caso el método println() está sobrecargado para poder aceptar como parámetro un objeto de tipo Date, encargándose el método de imprimir la fecha actual por la pantalla.
- Por otro lado, como parámetro de la función utilizamos un objeto creado mediante new, pero que no tiene asignado un handle. Es decir, va a ser un objeto temporal que, un cierto tiempo después de haber sido creado, será eliminado por el recolector de basura.

En las dos siguientes sentencias podemos comprobar el uso de una clase estándar para manejar cadenas constantes, llamada String. En primer lugar creamos el objeto mediante new, pasando como parámetro al constructor del objeto la cadena de caracteres que queremos manejar. El concepto de constructor será tratado ampliamente en el próximo capítulo así que, por ahora, nos basta con saber que es un método que contienen todas las clases y que facilita la inicialización de sus variables miembro. La clase String es un poco especial que nos permite usar una sintaxis similar a la del C++ para crear un objeto cadena:

String s2 = "Hola, de nuevo";

Las dos últimas sentencias del programa se limitan a llamar al método de clase prt con los argumentos apropiados (observar que no hemos necesitado crear ningún objeto Print).

CONCLUSIÓN

En este artículo de introducción hemos escrito poco código, pero asentado conceptos claves sobre la POO: encapsulación, polimorfismo, herencia, etc. También hemos comparado el concepto de handle con el de tipo básico, que nos ha servido para entender la gestión de memoria del lenguaje. Hemos revisado el concepto de alcance de un tipo de dato. Y por último hemos escrito un programilla simple para afianzar conocimientos.

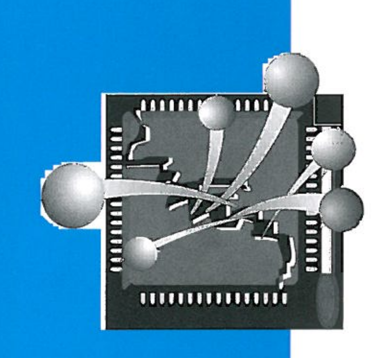
PC Adictos

A partir de Abril todas las semanas tú y tu Pc tenéis una cita en **LA 2**



COMUNICACIÓN OLE

Juan Manuel Menéndez



OLE se va imponiendo poco a poco como un estándar en la comunicación entre procesos. Con Visual Basic 4.0 el uso de OLE se ha fortalecido con la posibilidad de crear servidores OLE con este lenguaje

OLE (acrónimo de Object Linked and Embebbed) es un término que aparece con mucha frecuencia en revistas, artículos, libros, etc. Se podría decir que es una palabra que conviene utilizar de vez en cuando en el "mundillo" de los sistemas informáticos (amén de Java, Internet & Intranet, etc.). Básicamente OLE es un modelo conceptual que permite que unos programas accedan a los servicios suministrados por otros programas a través de una interfaz orientada a objetos. De una forma simplista se puede decir que *OLE es un mecanismo de comunicación entre programas.*

Esto puede parecer poco, pero sin embargo implica mucho; comunicar programas no es sólo pasar datos de un programa a otros, sino además:

- Determinar los programas que van a recibir los datos.
- Verificar si procede ejecutarlos.
- Sincronizar el traspaso de datos.
- Gestionar los mecanismos de traspaso.

Y todo ello mediante un interfaz de programación coherente, bien definido e independiente de cuales sean los programas involucrados en la comunicación.

Al ser interfaz orientada a objetos los suministradores de los servicios presentan a los peticionarios dos tipos de elementos:

- Propiedades
- Métodos

El modelo OLE está diseñado para ser implementado sobre diversas variedades de sistemas operativos. Cada implementación utilizará las características propias de cada sistema para la realización material del mismo. En el caso de Windows 95 el modelo OLE está en gran parte soportado por los siguientes mecanismos del sistema operativo:

- *Memory mapped files* (ficheros mapeados en memoria).
- Servicios RPC (*Remote Procedure Call*)

El objetivo de esta serie de artículos es ver la utilización de OLE desde Visual Basic 4 sin entrar en el modelo conceptual ni en la implementación real del mismo.

ESQUEMA DE LA COMUNICACIÓN OLE

En la comunicación OLE cada extremo de la misma juega un papel definido, uno de los extremos es el cliente OLE y el otro extremo es el servidor OLE. Vemos pues que el primer pilar de OLE es el de una arquitectura genérica Cliente-Servidor.

Una vez definidos los roles de cada elemento, veamos las formas de trabajar posibles en la interface:

- Activación del objeto y dejar que éste interaccione con el

usuario, tal y como hacen las aplicaciones contenedoras OLE. Mínimo conocimiento del servidor, mínimo control del mismo.

- Accediendo a métodos y propiedades de los servidores utilizando los mecanismos de automatización OLE. Mayor conocimiento y por tanto mayor control del servidor.

ARQUITECTURA DE LOS SERVIDORES OLE

Los servidores OLE son los encargados de realizar las peticiones de los clientes, para ello ponen a disposición de éstos sus métodos y propiedades.

Los servidores OLE pueden ser de dos tipos:

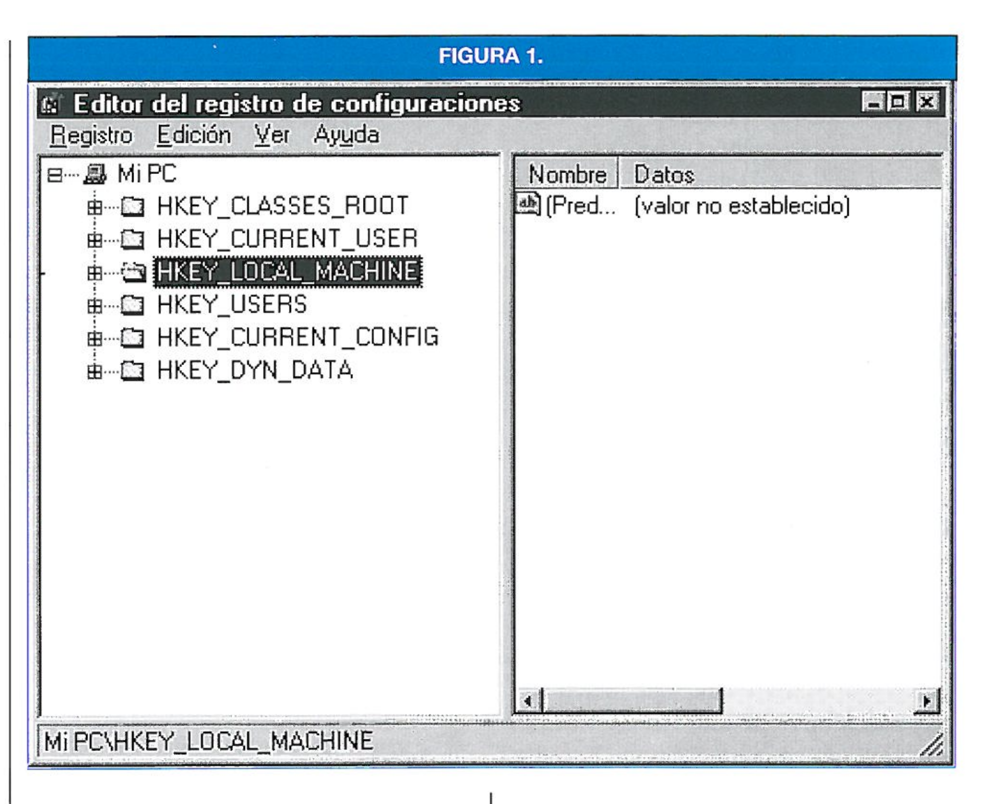
- In-process server: son los servidores que se ejecutan dentro del proceso cliente; se implementan con DLL's, por lo que su activación siempre es más rápida ya que evita toda la sobrecarga que supone para el sistema operativo la creación de un nuevo proceso.

- Cross-process server: son los servidores que se ejecutan como aplicación independiente. El trabajo extra que supone este modo de operar se ve compensada por la flexibilidad que aporta, al no estar compartiendo el espacio de direcciones del cliente, éste puede estar realizado en 16 bits, mientras el servidor puede ser de 32 bits; además el cliente no tiene porqué ejecutarse en el mismo equipo que el servidor.

A la hora de decidirse por un tipo de servidor, es necesario conocer bien el entorno sobre el que se va a implementar nuestro servidor, determinar si van a coexistir clientes en 16 o 32 bits (puede que se esté en una fase de transición) y ver las posibilidades de crear servidores en distintos equipos.

Como en cualquier tema informático, encontramos el dilema flexibilidad versus rapidez

FIGURA 1.



BASE DE DATOS OLE

Anteriormente se ha dicho que un mecanismo de comunicación entre programas que sea eficiente ha de conocer qué programa (servidor) ha de ser llamado cuando otro (cliente) requiere un servicio. Esta información en Windows está almacenada en el Registry.

El Registry es una monumental base de datos donde el Sistema Operativo guarda toda la información necesaria para funcionar. Para acceder a este fichero utilizaremos un programa suministrado por el sistema operativo, este programa es regedit.exe, que nos muestra al activarlo la ventana de la Figura 1. Toda la información que interesa a OLE está almacenada bajo el titulo HKEY_CLASSESS_ROOT.

El servidor OLE que vamos a investigar a través del Registry es el procesador de texto Word.6.

La expansión del árbol HKEY_CLASSESS_ROOT tiene las entradas divididas en dos grandes grupos:

- Las extensiones de los nombres de los ficheros a las cuales hemos asociado algún programa.

- Los diversos servidores OLE que tenemos instalados en nuestro sistema.

La forma más sencilla de localizar cuál es el nombre de nuestro servidor OLE es seleccionar la entrada relativa a la extensión asociada a nuestro programa, en el caso de Word, la extensión inicial asociada es .DOC. Si no conocemos la extensión asociada al programa lo más fácil es ejecutarlo y abrir un fichero, el cuadro del diálogo nos muestra ya seleccionada cuál es la extensión asociada.

Abriendo la carpeta .DOC (Figura 2) el valor que encontramos es **Word.Document.6**, buscamos este valor a continuación en el mismo nivel del jerarquía del árbol en el Registry, expandimos las entradas y

abrimos la carpeta CLSID (Figura 3); vemos el Identificador que el sistema asigna al servidor OLE.

El CLSID es una clave única de 128 bits que identifica a cada objeto. Esta clave es creada por la función

CoCreateGUID

que está implementada en la DLL gestora de OLE (OLE32.DLL o OLE2.DLL). Esta función garantiza que la CLSID será única en nuestro equipo. Todos los programas que se instalan en nuestro equipo graban su CLSID en el Registry.

Sin cambiar de jerarquía, buscamos la entrada CLSID y la expandimos; en ella vemos todos los identificadores que hay definidos. Buscamos la nuestra (para ello podemos utilizar el buscador del programa). La entrada expandida contiene diversas carpetas donde se guarda toda la información relativa al servidor (Figura 4).

Entre esta información destacamos la carpeta:

INPROCHANDLER

La carpeta InProcHandler nos dice cuál va a ser la DLL que va a gestionar nuestras llamadas OLE. Windows nos ofrece dos valores: OLE2.DLL y OLE32.DLL (para 16 y 32 bits). Cuando invocamos a un método de un servidor OLE hay dos grandes diferencias con respecto a cuando invocamos a una función normal de una DLL:

- La sintaxis; en una función es:

función(parámetro 1,...., parámetro n).

Si lo hacemos en un objeto es:

nombre_del_objeto.método(parámetro 1, ..., parámetro n)

- La definición: los programas necesitan informar al sistema de la DLL en que van a encontrar la función, para que éste puede buscarla cuando sea necesario. En el caso de OLE la única función necesaria es la referenciada en el parámetro InProcHandler y ésta se encarga de localizar el resto de la información.

Con la versión 4 de VB ya es posible la creación de servidores OLE tanto in-proccess como cross-proccess

Como podemos ver no importa cuántas DLL's servidores OLE sean necesarias, desde el cliente OLE la única DLL que hay de referenciar es la InprocHandler. Y podemos olvidarnos de la sentencia Declare.

La única información que nos falta para completar el sistema es saber qué programa o DLL va a ser cargado cuando requiramos los servicios del servidor OLE, esta información viene en una de las siguientes carpetas:

- LocalServer32: que nos indica que es un servidor cross-proccess de 32 bits.
- LocalServer: que nos indica que es un servidor cross-proccess de 16 bits.
- InProcServer: que nos indica que es un servidor in-proccess (32 bits).
- InProcServer32: que nos indica que es un servidor in-proccess (16 bits).

FIGURA 3.

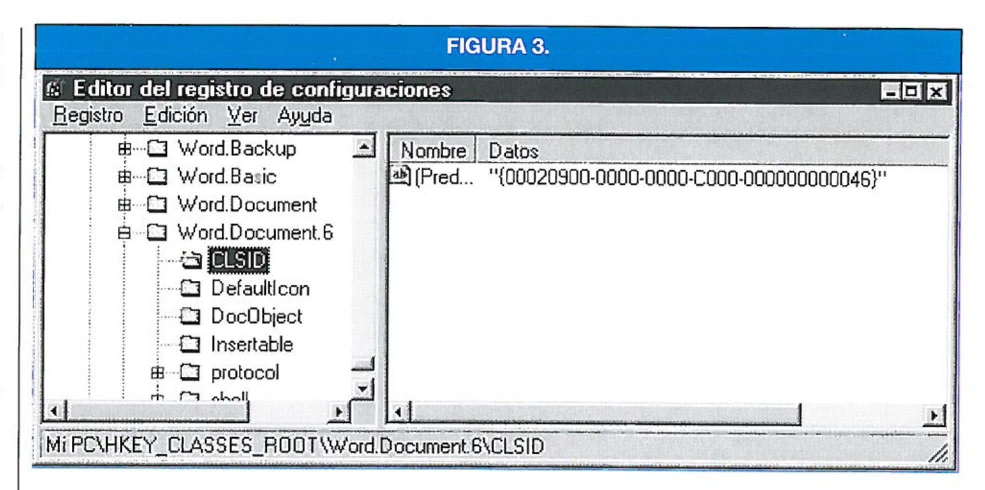
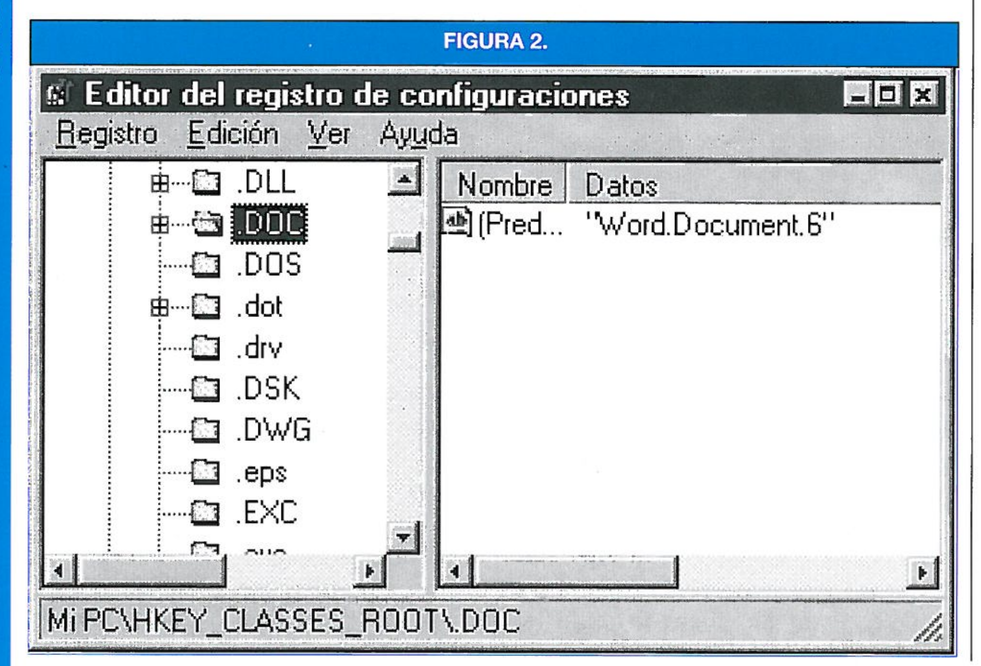


FIGURA 2.

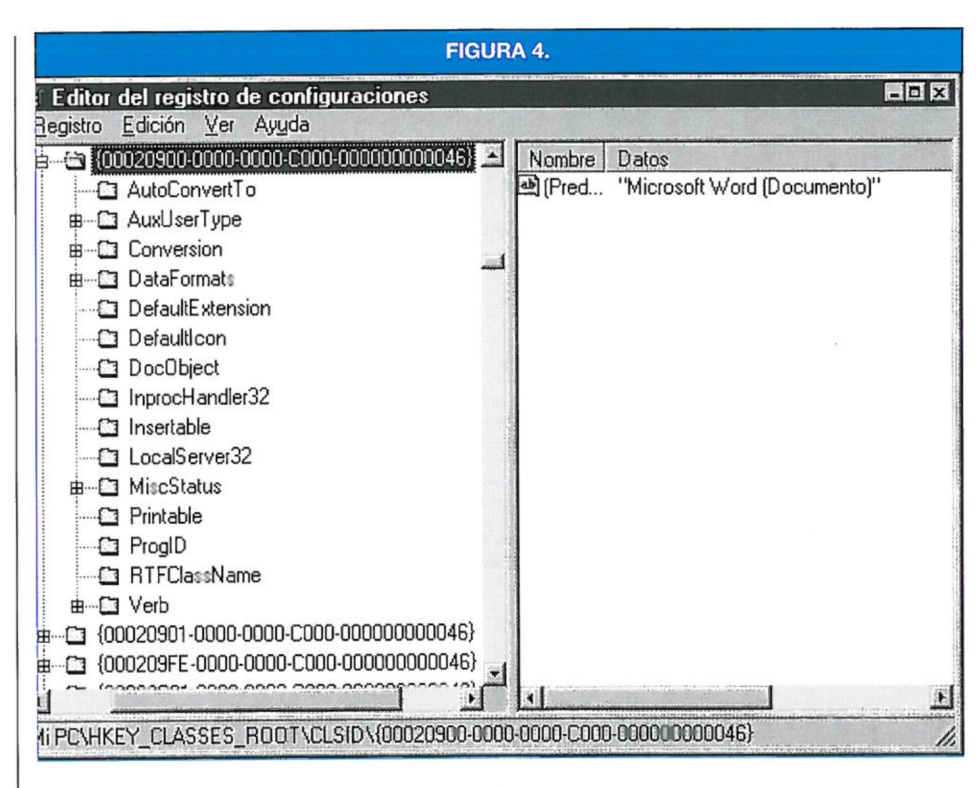


Hay que notar que un mismo servidor OLE puede tener carpetas para 16 y 32 bits bajo el mismo CLSID.

En el caso de Word.6 la carpeta que aparece es LocalServer32 que nos indica que el servidor es un cross-proccess de 32 bits y el contenido de la carpeta es el path completo de WINWORD.EXE.

Como información adicional, en la carpeta ProgID está el nombre del servidor OLE dentro de HKEY_CLASSESS_ROOT con lo que el círculo de información dentro del Registry queda cerrado.

FIGURA 4.



VISUAL BASIC Y OLE

Hasta la aparición de Visual Basic 4, este sistema de programación se hallaba limitado a ser un cliente OLE. Las aplicaciones OLE que ahora es posible crear con VB 4 son:

- Aplicaciones contenedoras OLE. Son la forma más sencilla de trabajar con OLE puesto que sólo se limitan a activar el servidor y dejar que éste trabaje de forma autónoma con un tercer agente (normalmente el usuario en la pantalla).
- Aplicaciones clientes usando Automatización OLE. Este tipo de aplicaciones tiene un control total sobre los servidores pero requieren un conocimiento sobre las propiedades y métodos del servidor sobre el que se está trabajando.
- Aplicaciones servidoras OLE (inproccess y cross-proccess). Los servidores OLE deben de establecer los métodos y propiedades de las que deben de hacer uso los clientes.

CREACIÓN DE UNA APLICACIÓN CONTENEDORA OLE EN VB

Como anteriormente se ha dicho, una aplicación contenedora OLE apenas necesita conocer el servidor OLE sobre el que actúa.

Una aplicación contenedora OLE puede funcionar casi sin código, sólo actuando con los propiedades del objeto en tiempo de desarrollo, hasta conseguir que la aplicación se comporte de la manera deseada, o bien utilizar las propiedades y métodos del contenedor OLE para conseguir a través de ellos un mayor control.

Para crear una aplicación de este tipo con VB 4 (y VB 3), una vez creado un proyecto nuevo en VB seleccionamos de la caja de herramientas el control OLE y lo llevamos al form.

Las modificaciones en un control OLE incrustado se mantienen durante la sesión, perdiéndose al cerrarla

Nos aparece un cuadro de diálogo (Figura 6), cuya información está claramente sacada del Registry como hemos visto antes. Seleccionamos Microsoft Word (Documento) que es el contenido de la carpeta Word.Document.6 en HKEY_CLASSESS_ROOT. La check box presentar icono nos da la opción de presentar nuestro objeto como icono; para ver mejor la diferencia entre ambas presentaciones, seleccionamos presentar icono y al lado creamos otro contenedor del mismo tipo con esta opción desactivada.

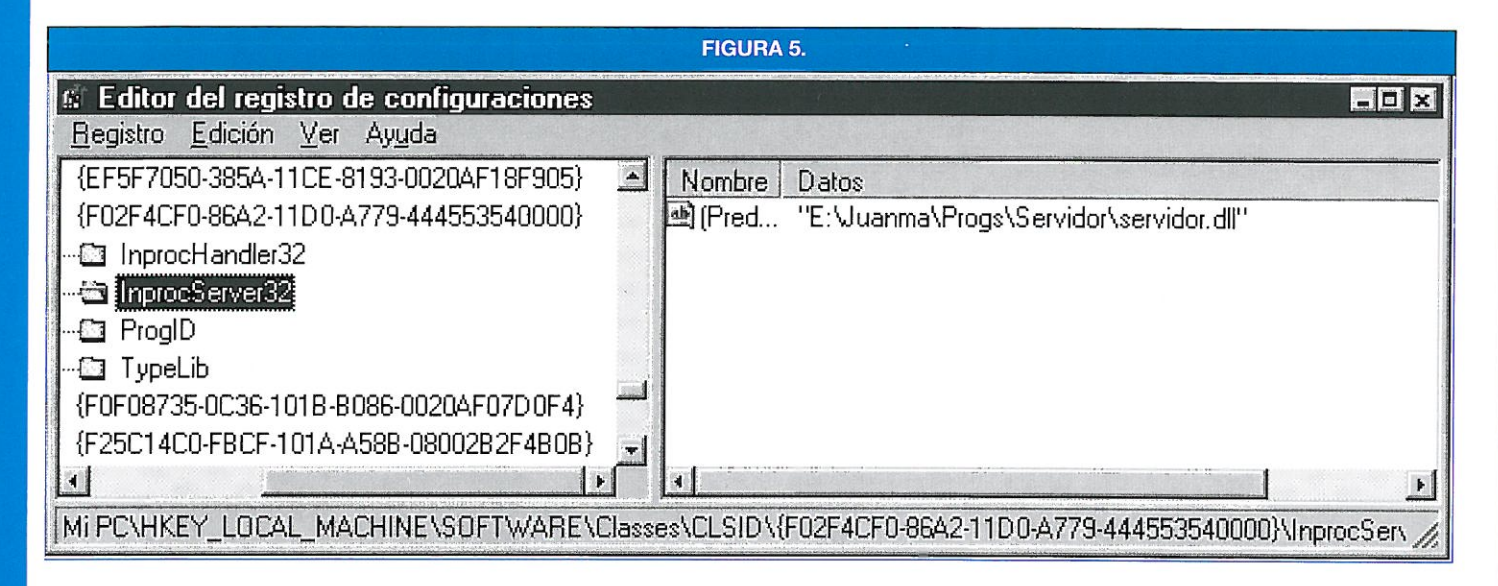
Con cada creación del objeto, Word se abre, por lo que deberemos cerrarlar para seguir trabajando. Nos situamos en cualquiera de los objetos y abrimos el menú contextual (pulsando el botón derecho del ratón), en las últimas posiciones hay dos opciones: Editar y Abrir a las que se hacen referencia dentro del CLSID en la carpeta Verb del Registry.

Veamos en la ventana de propiedades, algunas referentes al control.

1) Autoactivate. Nos indica la forma en que se va a activar el objeto, tiene cuatro valores:

- 0 Manual. El objeto solamente se activa a través del programa.
- 1 Enfoque. Se activa al obtener el enfoque. (No todos los servidores tienen esta opción).

FIGURA 5.



- 2 Doble click. Se activa cuando el usuario hace doble click en el control.
- 3 Automático. La aplicación se activa según el método de activación normal, cuando el control recibe el enfoque o cuando el usuario hace doble click.

Word no tiene activación con el enfoque, por lo que dejaremos el valor 2, que es el predeterminado.

2) Class. Esta propiedad indica el nombre de la clase del objeto.

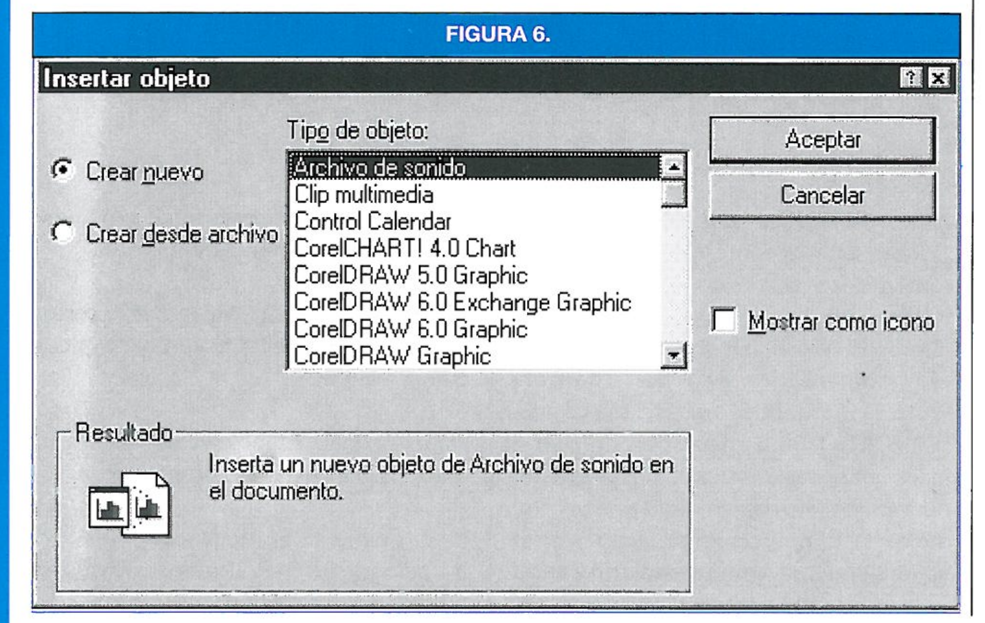
3) OLEType: Tipo del objeto, tiene 3 valores:

- 0 Enlazado.
- 1 Incrustado.
- 2 Cualquiera de las dos.

4) SourceDoc: Es el nombre del objeto que queremos enlazar o incrustar. En esta propiedad podemos seleccionar el nombre del fichero que queremos incrustar o vincular.

A continuación podemos ver los diferentes comportamientos que muestran nuestros controles en ejecución.

FIGURA 6.



APARIENCIA

Si nos centramos en el control visto como icono y en la propiedad SourceDoc seleccionamos un documento Word vinculado (marcando la check box correspondiente). En la ejecución de nuestro programa vemos el icono de Word con la pequeña indicación a la izquierda, indicando que es un acceso directo. Si activamos el control (con doble click), se nos abre una ventana con Word y el documento seleccionado. Si abrimos el menú Archivo, vemos que éste es igual al que tiene la aplicación cuando se arranca desde el escritorio.

OLE es un mecanismo de comunicación entre programas

Pongamos ahora el documento como incrustado y arranquemos la aplicación. En este caso, en el icono de Word ha desaparecido la flecha de acceso directo. Activando el control se arranca Word. En el menú de archivo vemos, no obstante, algunas diferencias en las opciones que se refieren al "salvado" del documento.

Un comportamiento similar, aunque en diferente formato, tenemos con el control cuya presentación es de contenido.

COMPORTAMIENTO

Si el objeto activado es incrustado y si cerramos actualizando el contenido, éste queda en el objeto, como se puede observar si el objeto está en el

form en formato de contenido, o bien volviendo a activar el objeto si la representación es de icono.

Si cerramos la aplicación y la volvemos a abrir, vemos que todas las modificaciones que hemos realizado sobre el objeto se han perdido. Podemos guardar las modificaciones realizadas si seleccionamos la opción Guardar copia como... pero la copia se realizará sobre un fichero que quedará totalmente desligado de nuestra aplicación.

Por el contrario, en el caso de que el objeto seleccionado fuera vinculado las modificaciones quedarían sobre el fichero vinculado y éstas permanecerían a lo largo de las diversas sesiones que se mantuvieran con la aplicación. De la misma forma, cualquier modificación sobre el documento que se realiza de forma externa a la aplicación, desde otra aplicación contenedora o desde el propio Word, aparecerá reflejada en el objeto cuando se invoque a través de la aplicación.

Las modificaciones en un control OLE incrustado se mantienen durante la sesión, perdiéndose al cerrarla

En la tabla 1 podemos ver la diferencia de tamaño del módulo ejecutable según el objeto sea incrustado o vinculado, partiendo de un fichero Word de 24 Kb de tamaño. Se puede ver la diferencia de tamaño debida a la inclusión del objeto en el propio contenedor.

Tabla 1.- Tamaño del módulo

	Vinculado	Incrustado
EXE	15	34

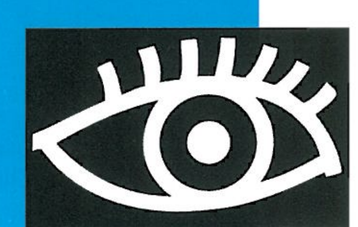
CONCLUSIONES

Hemos visto cómo se puede crear una aplicación contenedora OLE sin necesidad de conocer las características del servidor y sin programación, únicamente jugando con las propiedades del control en tiempo de diseño, hasta conseguir el resultado deseado. Esta es la forma más simple de trabajar con esta interfaz.

A lo largo de los siguientes artículos veremos cómo potenciar el uso de las aplicaciones contenedoras OLE utilizando código, cómo crear aplicaciones clientes a través de Automatización OLE, para acabar creando nuestros propios servidores en Visual Basic, y cómo éstos pueden, a su vez, actuar de clientes de otros servidores OLE.

BORLAND C++ BUILDER

Arsenio Molinero Puente



¿Quién dijo que programar en C++ era difícil?. Siempre se ha pensado que la curva de aprendizaje del C++ es tremendamente cerrada, pero ahora ha llegado el momento de quitar las curvas de la carretera y poner rectas, ha llegado el momento del Borland C++ Builder. Este producto es tan fácil de manejar que se pueden hacer programas simplemente pinchando con el ratón, sin tener que recurrir al teclado.

El Borland C++ Builder no es una herramienta Visual de programación normal y corriente, posiblemente este producto sea el primer entorno de desarrollo Visual para C++ cien por cien orientado a Objetos. El manejo del entorno es similar al del Visual Basic y se hace notar la mayor potencia del C++. La primera sensación que te produce es la de poder realizar cualquier cosa sin ninguna limitación, de tener en tus manos un todo terreno del París-Dakar, que se adapta a cualquier tipo de situación y a la vez es rápido.

Este producto es la herramienta ideal para aprender a programar en C++ para Windows, pero no por ello carece de las herramientas más flexibles y potentes del entorno. El Borland C++ Builder hará las delicias de expertos programadores, pues facilita la gestión de las clases e incluye infinidad de librerías para cualquier tipo de desarrollo, permitiendo ahorrar tiempo en la fase de creación de controles y así poder utilizarlo a la hora de comenzar el desarrollo real de la aplicación.

Borland C++ Builder es el hermano gemelo del *Delphi*, incluso va más allá que este ya que permite escribir código en lenguaje Pascal, en Ensamblador y en C++. Puede gestionar ficheros objetos (.OBJ) de cualquiera de los lenguajes antes citados y también incluir el código fuente en el formato original.

La librería *VCL* (*Visual Component Library*: Librería de Componentes Visuales) es idéntica a la que trae el entorno de Delphi. Esta librería es utilizada para la creación de las Formas y los Objetos. En el Borland C++ Builder se proporciona el código fuente de las funciones de la librería y tiene la peculiaridad de estar escrito en Pascal.

En la Figura 1 podemos observar el aspecto del programa, cuya apariencia es idéntica a la de Delphi. Nos encontramos con una ventana de gestión de eventos y propiedades y una barra de herramientas donde se pueden encontrar todo tipo de objetos que podemos incrustar en las formas, concepto que se explica más adelante.

DISTRIBUCIÓN DEL PRODUCTO

Existen tres versiones del programa:

- C++ Builder Standard. Esta es la versión para usuarios que necesiten crear

CUADRO 1.

```
//-----------------------------------------------------------
void __fastcall TForm1::FormMouseMove(TObject *Sender,
                                      TShiftState Shift,
                                      int X, int Y)
{
|
}
//-----------------------------------------------------------
```

CUADRO 2.

```
//---------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
OpenDialog1->Execute();
Image1->Picture->LoadFromFile(OpenDialog1->FileName);
}
//---------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------
```

aplicaciones para *Windows 95 y Windows NT* sin necesidad de realizar complejos accesos a bases de datos ni opciones muy avanzadas. Este paquete trae todo lo necesarios para realizar cualquier tipo de aplicación.

- C++ Builder Cliente/Servidor. Herramienta ideal para realizar aplicaciones de interconexión entre distintas máquinas. Este paquete es casi igual al profesional , menos en la parte de bases de datos, en esa parte esta mejor equipado.

- C++ Builder Profesional. Esta es la más avanzada de todas, trae los mismos componentes que las anteriores y toda la parte de cliente *SQL, ODBC.* También trae un conjunto de utilidades interesantes de las que podemos destacar el *InstallShield* para la creación de paquetes de instalación, una serie de controles y clases para gestionar *TCP/IP* y poder hacer aplicaciones de *Internet, el famoso ActiveX* y el código fuente de la *VCL.*

DESARROLLO EN BORLAND C++ BUILDER

En Borland C++ Builder trabajaremos normalmente con proyectos, estos proyectos pueden contener módulos Objetos, código y formas. Las formas son las ventanas donde colocamos los controles o, dicho de otra manera, las formas son el interfaz entre la aplicación y el usuario. El concepto de formas se ha heredado de Visual Basic, de hecho, la manera de trabajar es muy similar.

Todos los controles y formas que podemos gestionar son objetos, en los documentos de ayuda del producto encontraremos referencia a todas las clases de esos objetos con una explicación clara de los miembros y ejemplos de cómo se deben utilizar. Éste es uno de los aspectos que más me han llamado la atención a la hora de realizar el estudio del producto, la documentación que le acompaña es excepcional.

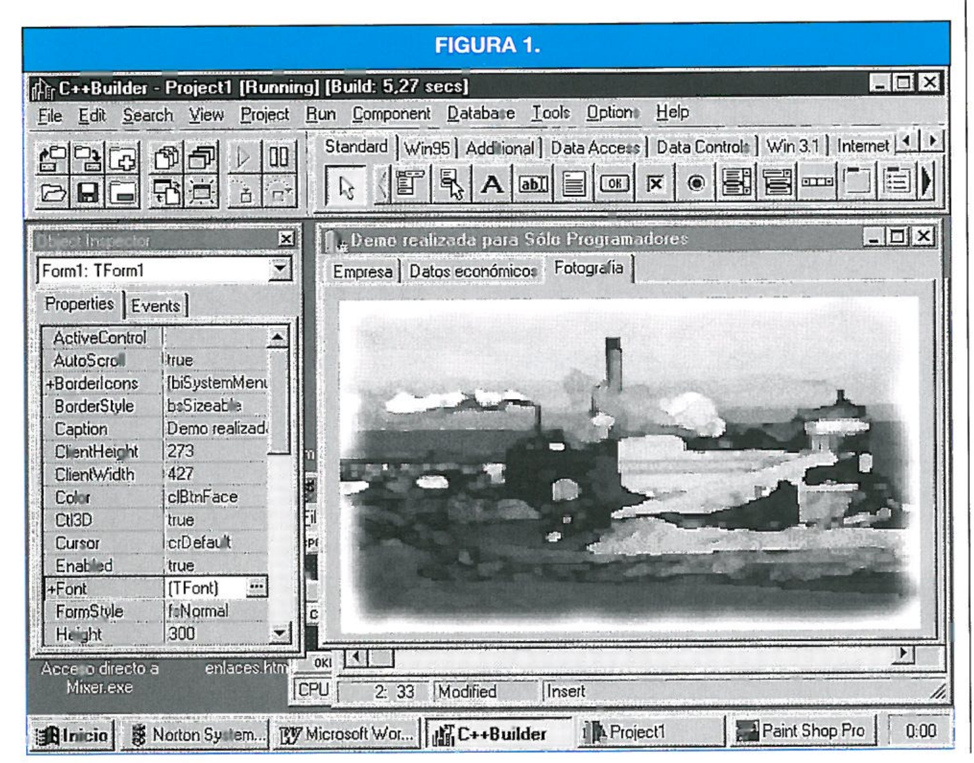
Para poder colocar objetos en las formas disponemos de la barra de herramientas donde los objetos están organizados por grupos. Tenemos un total de 107 objetos divididos en 11 grupos. Esto no significa que ahí se acabe todo, pues también se pueden realizar llamadas a las distintas *API'S* y controles. A continuación podemos ver los distintos grupos que existen:

- "*Standard*". En la Figura 2 podemos observar el aspecto de este grupo, en él encontramos los objetos típicos de Windows: MainMenu, PopUp, Label, Edit, Memo, Button, CheckBox, RadioButton, ListBox, ComboBox, ScrollBar, GroupBox, RadioGroup y Panel. Éstos son los heredados de Windows 3.X.

- "*Win95*". Este grupo tiene los objetos nuevos de Windows 95, en la Figura 3 podemos observar los objetos del grupo, éstos son, por orden de visualización: TabControl, PageControl, TreeView, ListView, ImageList, HeaderControl, RichEdit, StatusBar, TrackBar, ProgressBas, UpDown y HotKey.

- *"Additional"*. Aquí podemos encontrar algunos controles muy útiles que se suelen echar en falta en el estándar de Windows, estos controles son: BitBtn, SpeedBotton,

FIGURA 1.



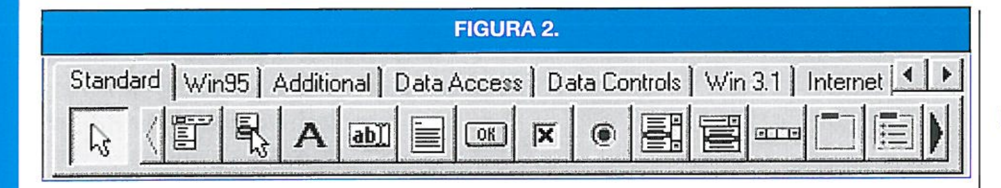


FIGURA 2.

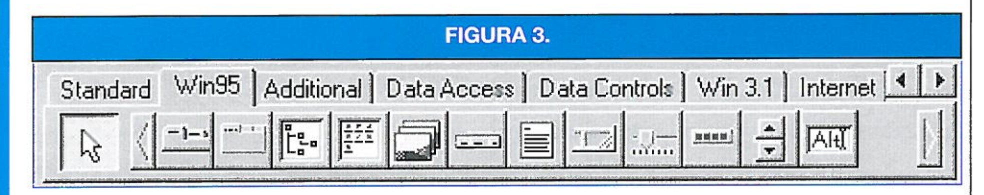


FIGURA 3.

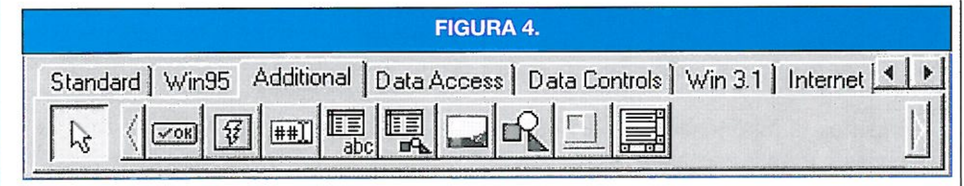


FIGURA 4.

MaskEdit, StringGrid, DrawGrid, Image, Shape, Bevel, ScrollBox. Podemos ver el aspecto de este grupo en la Figura 4.

- *"DataAccess"*. Es el grupo que contiene objetos para el control de la accesibilidad a bases de datos, en la Figura 5 se pueden observar los controles siguientes: DataSource, Table, Query, StoredProc, DataBase, Session, BatchMove, UpdateSQL.

- *"DataControls"*. Este grupo está ligado al anterior, en él vamos a encontrar los controles necesarios para gestionar campos y demás, los controles son los siguientes: DBGrid, DBNavigator, DBText, DBEdit, DBMemo, DBImage, DBListBox, DBComboBox, DBCheckBox, DBRadioGroup, DBLockUpListBox, DBLockUpComboBox y DBCtrlGrid. Se pueden ver en la Figura 6.

- *"Win 3.1"*. Los controles que tenemos en este grupo son muy interesantes, son controles que estamos acostumbrados a ver en las aplicaciones comerciales. En la Figura 7 se pueden identificar de izquierda a derecha los siguientes: DBLockUpList, DBLockUpCombo, TabSet, OutLine, Header, TabbedNoteBook y NoteBook.

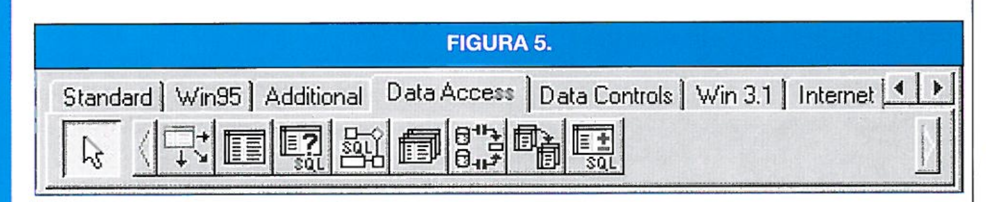


FIGURA 5.

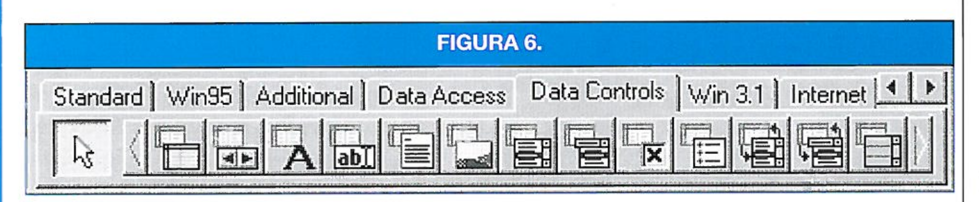


FIGURA 6.

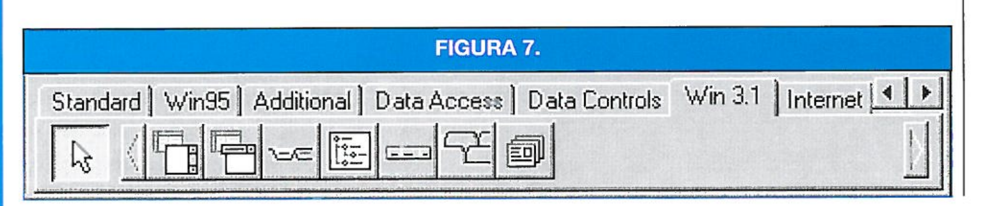


FIGURA 7.

- *"Internet"*. Este es uno de los grupos más sorprendentes, con él podemos realizar todo tipo de operaciones necesarias de TCP/IP, desde correo, FTP, Web, y News, con él puedes crear tus propias aplicaciones cliente-servidor para Internet. Los componentes de este grupo son: FTP, HTML, HTTP, NNTP, POP, SMTP, TCP, UDP.

Como se puede ver, el paquete es muy completo y trae el API de Winsock. En la Figura 8 se ve el aspecto de este grupo.

- *"Dialogs"*. Herramienta imprescindible a la hora de manejar los diálogos estándar. En este grupo están los objetos encargados de realizar las llamadas a los diálogos estándar, abrir ficheros, guardarlos, imprimir, etc... En la Figura 9 se puede ver este grupo, sus componentes son: OpenDialog, SaveDialog, FontDialog, ColorDialog, PrintDialog, PrinterSetupDialog, FindDialog y ReplaceDialog.

- *"System"*. Existen objetos para manejar las funciones más utilizadas del sistema de Windows, en la Figura 10 tenemos el grupo System el cual está compuesto de los controles: Timer, PaintBox, FileListBox, DirectoryListBox, DriverComboBox, FilterComboBox, MediaPlayer, OleContainer, DDEClientConv, DDEClientItem, DDEServerConv y DDEServerItem.

- *"QReport"*. La Figura 11 contiene los objetos necesarios para realizar informes, estos objetos son: QuickReport, QRBand, QRGroup, QRDetailLink, QRLabel, QRMemo, QRDBText, QRDBCalc, QRSysData, QRShape y QRPreview.

- *"ActiveX"*. Impresionante, este grupo se merece un diez, tiene gestión de gráficos de barras, hoja de cálculo y diccionario. Los Objetos que aparecen en la Figura 12 son los siguientes: ChartFX, VCFirstImpression, VCFormulaOne, VCSpeller y GraphicsServer.

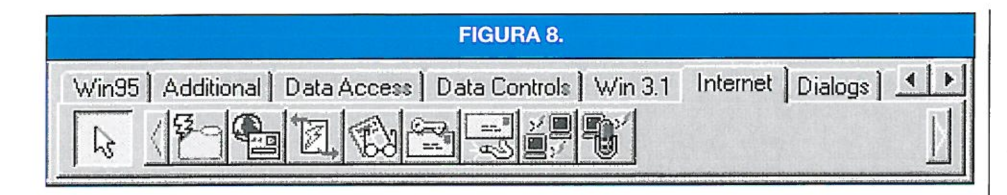


FIGURA 8.

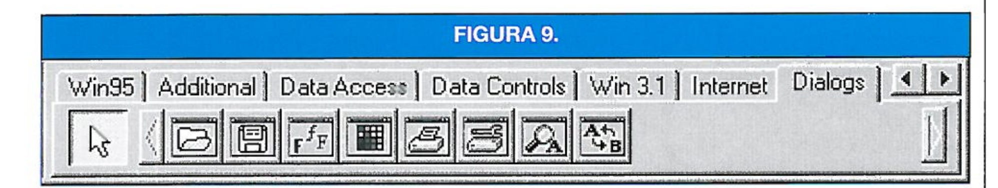


FIGURA 9.

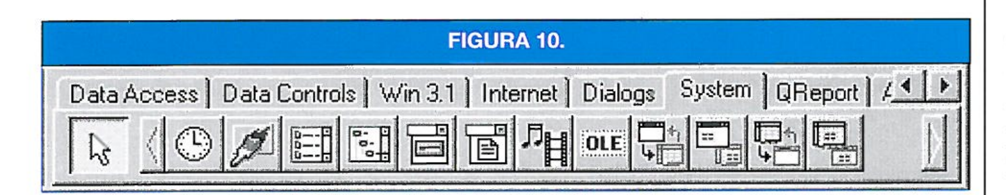


FIGURA 10.

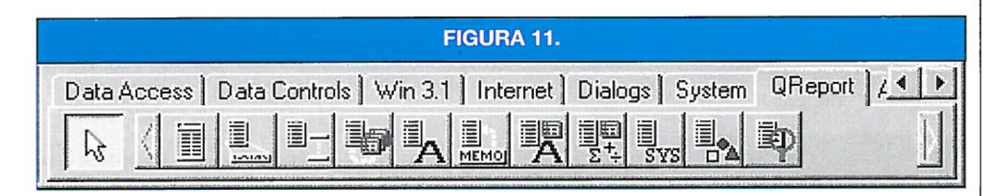


FIGURA 11.

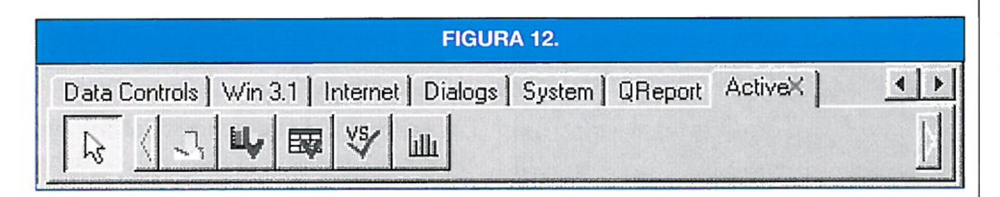


FIGURA 12.

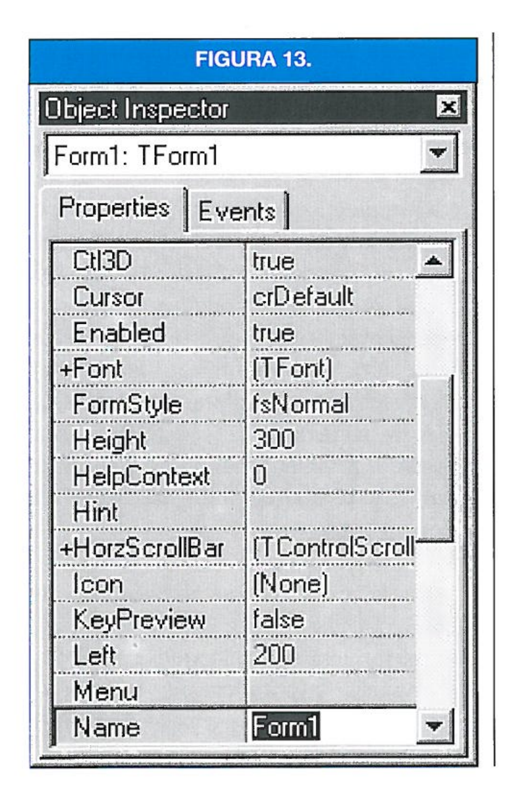


FIGURA 13.

Después de colocar los objetos, tendríamos que configurarlos y crear los eventos necesarios. Para realizar estas operaciones existe la ventana Object Inspector. Esta ventana está dividida en tres partes, la Lista de Objetos creados, las Propiedades y los Eventos, en la Figura 13 podemos ver la Lista de objetos y las Propiedades. Cuando queremos ver los Eventos o Propiedades de los objetos, tenemos que pulsar en la lista y seleccionar el objeto a tratar; otra manera sería pulsar con el ratón encima del objeto. La ventana de propiedades te permite cambiar el tipo de letra, el color, los elementos del objeto y muchas otras características del objeto. En la Figura 14 podemos ver la misma ventana pero con la lengüeta de Eventos seleccionada. Para poder manejar cualquier evento del objeto, se puede hacer un doble click con el ratón en el evento deseado. Como se puede observar el número de eventos disponibles es mucho mayor que el que nos proporciona Visual Basic.

En el Cuadro 1 podemos ver el código que genera el Borland C++ Builder cuando realizamos un doble click sobre el evento MouseMove. A partir de ese momento podemos escribir el código necesario para ese evento.

MANOS A LA OBRA

Después de hablar tanto del producto, ha llegado la hora de ponernos a hacer una pequeña demostración.

La aplicación que vamos ha crear es un visualizador de gráficos sencillo. Este programa tiene un recuadro donde visualizará las imágenes que seleccionemos.

La aplicación está compuesta por una forma con 4 objetos. El primer objeto que ponemos es un Image para poder visualizar la imagen, después ponemos dos botones, el primero es para pedir el fichero a visualizar y el segundo para finalizar la ejecución de la aplicación. Después tenemos un objeto del tipo OpenDialog que es el que utilizamos para que aparezca el diálogo estándar de abrir fichero, con éste permitimos al usuario seleccionar cualquier archivo gráfico. En la Figura 15 podemos ver el aspecto de la forma creada, también podemos ver la lista de objetos desplegada donde aparecen los objetos creados hasta el momento.

Borland C++ Builder permite escribir código en Pascal, en ensamblador y en C++ y gestionar ficheros.obj

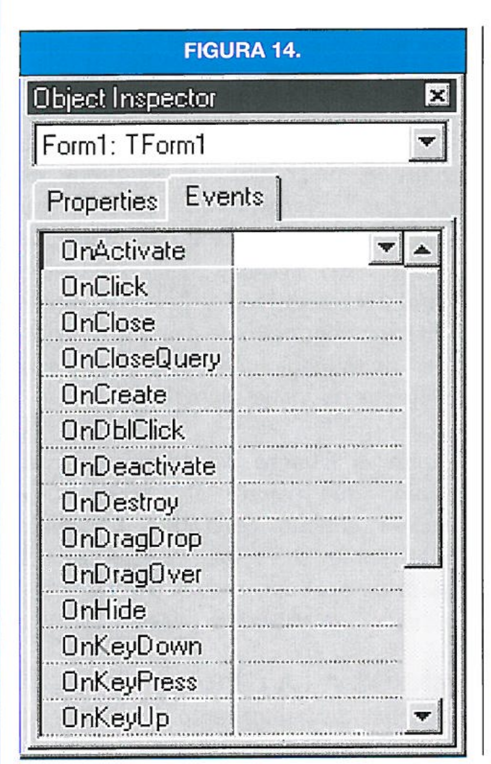


FIGURA 14.

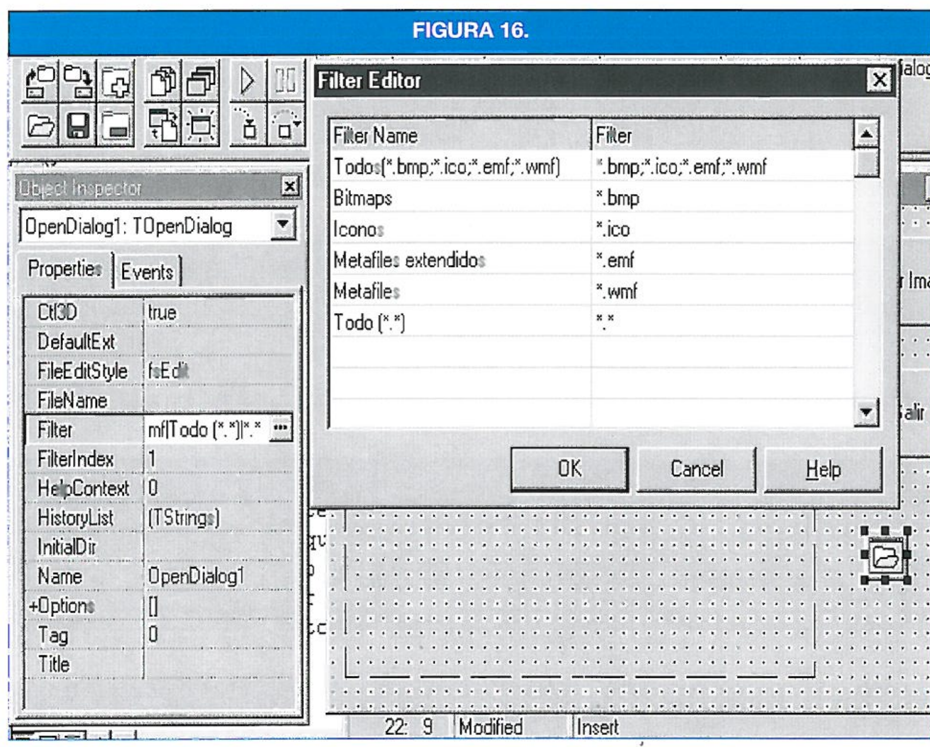


FIGURA 16.

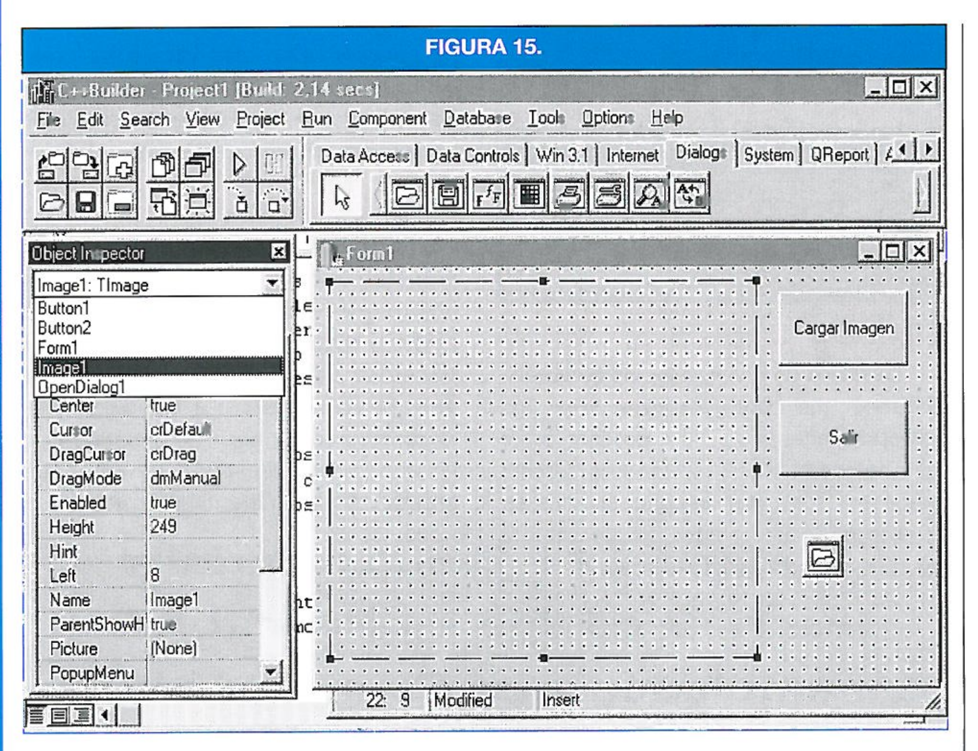


FIGURA 15.

El Objeto OpenDialog necesita que le configuremos la propiedad Filter para que permita visualizar sólo imagenes permitidas. En la Figura 16 podemos ver la ventana Filter Editor que aparece cuando pinchamos en Filter.

Tras hacer esos cambios sólo nos hace falta pinchar en el Button1 para que llame al OpenDialog y cargue la imagen seleccionada y después pinchar en el Button2 para poner el comando necesario para que finalice la aplicación, en el Cuadro 2 podemos ver el código escrito para que realice esas funciones.

Ya tenemos un visualizador de gráficos sencillo. El compilado y enlazado de esta aplicación es muy rápido, aproximadamente un segundo y medio. En la Figura 17 aparece la ventana de información en la que podemos ver el tamaño de la aplicación y las líneas de código que ha compilado.

CONCLUSIONES

Ha aparecido un duro competidor del Microsoft Visual C++. El visual C++ se ha hecho con el mercado de entornos de desarrollo en C++, no sólo a nivel de empresas de desarrollo, también a nivel de usuarios. El Borland C++ Builder va a abrirse paso rápidamente, pues es un entorno muy manejable, estable y rápido.

Otra de las características por la que merece la pena conocer el entorno es por la terminación final del producto, en la que juega un papel importante la documentación que aporta. Esta documentación (totalmente en inglés) es una de las más completas que he visto, Solo he detectado un pequeño error: cuando compilas y tienes errores en el código no te aparecen los mensajes del error a simple vista.

En la parte de la depuración de código, Borland sigue dando las mejores herramientas, en lo que a facilidad de manejo y potencia se refiere; el depurador sigue en la misma línea que las versiones anteriores.

GENERACIÓN DE NÚMEROS ALEATORIOS

Francisco José Abad

Desde tiempos remotos, el hombre ha sentido una atracción innata por temas como el futuro, el destino, el azar,... Como una forma de desafiar ese ente insondable aparecieron innumerables juegos de azar que han acompañado al ser humano ya sea como divertimento ya sea como negocio, y que aparecen en cualquier libro de historia. Los matemáticos antiguos ya se interesaron en el estudio de este tipo de juegos y probablemente a esto debamos hoy en día el cálculo de probabilidades o la estadística.

En este artículo se tratan distintas técnicas para conseguir una sucesión de números aleatorios. A pesar de lo que se pudiera pensar en un primer momento sobre ellos, no sirven sólo, ni mucho menos, como pasatiempo de sesudos matemáticos ni para programar juegos de azar. Tiene aplicaciones extensísimas en campos como la simulación de todo tipo, por ejemplo industrial, de resistencia de materiales, estructuras, etc., en resolución de problemas matemáticos (por el llamado método de Montecarlo), cálculo de áreas e integrales, métodos de optimización, resolución de sistemas lineales, etc. Sin duda alguna, todas las ciencias que intentan modelizar la realidad para su estudio (el 99% de ellas) tienen en esta teoría la herramienta indispensable de su trabajo.

En un primer momento, para generar números aleatorios se usaban métodos manuales, ya sea lanzamiento de dados, cartas, etc. Más tarde se crearon artilugios mecánicos o eléctricos dedicados simplemente a generar secuencias de números. Uno de estos aparatos consistía en un disco que se hacía girar y según donde se parase generaba una cifra u otra (algo parecido a una ruleta). Otros métodos más curiosos consistían en elegir números de teléfono de una guía u obtener cientos de miles de decimales del número *pi*. El problema de la mayoría de estos métodos es la baja reproducibilidad del experimento, es decir, muy difícilmente se podría repetir una secuencia de lanzamiento de dados, por ejemplo. Este problema hace difícil comprobar que los números generados cumplan unas determinadas propiedades que aseguren su aleatoriedad. Debido a esto, se construyeron enormes tablas de números aleatorios ya contrastadas (algunas eran millonarias) que se podían usar en las aplicaciones que fueran necesarias. Con la aparición de los ordenadores desapareció la necesidad de mantener esas tablas.

El problema al que se tuvieron que enfrentar los programadores fue cómo hacer que una máquina secuencial, determinista y tonta 'eligiese' un número al azar. Un conocedor del ordenador podría elegir un elemento de la máquina que cambia muy rápidamente y que, a priori, es imprevisible: el reloj. Consultando el reloj en un momento dado podría obtenerse un número perfectamente aleatorio. El problema se presenta cuando se necesita más de un número. En este momento, si leemos repetidamente el reloj, en el mejor de los casos obtendremos una serie ascendente de números y en el peor de los casos, cuando se necesitan muchos números



En este artículo se trata de algo tan interesante como la manera de programar con un ordenador el azar.

rápidamente, el mismo número repetido. Está claro que los números que se obtienen no son aleatorios.

Llegado este punto, se ha de definir lo que es una serie aleatoria. Para que una serie se considere aleatoria, debe cumplir un conjunto de propiedades:

- debe ser uniformemente distribuida en su rango, es decir, que no se repita un conjunto de cifras muchas veces más que otro. Si se disponen en una recta todos los números generados, no deben aparecer huecos grandes.
- no se debe apreciar una dependencia entre los valores; por ejemplo, la serie 1, 2, 3, 4... es uniformemente distribuida, pero está claro que su aleatoriedad es muy baja.

Respecto a esto, conviene puntualizar un aspecto, que se verá con un ejemplo: se lanza un dado cinco veces y otro otras cinco, con el siguiente resultado

Dado A: 3, 1, 5, 2, 6

Dado B: 6, 6, 6, 6, 6

Está claro que el segundo dado parece trucado y que la serie aleatoria es la primera, pero debemos tener en cuenta que, con un dado perfecto, ambas series tienen la misma probabilidad de aparición ((1/6)*(1/6)*(1/6)*(1/6)*(1/6)), por lo tanto, las dos series tienen el mismo grado de aleatoriedad y no podemos decir que una tirada haya sido mas aleatoria que la otra. ¿Donde está el truco?, pues está en que, en una serie *infinita* de tiradas el número de apariciones de cada cara debería ser muy parecido. Por lo tanto, en una serie aleatoria que tienda a infinito la cantidad de apariciones de cada número generado deberá ser muy parecida, pero puede darse el caso de subseries como la del dado B.

Los métodos actuales de generación de números aleatorios se basan en algoritmos recursivos que basan cada número en una función que depende de uno o varios predecesores. El primero de estos métodos fue presentado por von Neumann y Metropolis en 1940, el llamado método del centro de los cuadrados (midsquares method), y consiste en:

paso número	Z_i	U_i	Z_i^2
0	7182	-	51581124
1	5811	0.5811	33767721
2	7677	0.7677	58936329
3	9363	0.9363	87665769
4	6657	0.6657	44315649
5	3156	0.3156	09960336
...	...	...	...

Figura 1. Ejemplo del algoritmo de los midsquares.

1.- *Elegir un entero de 4 cifras Z_0*

2.- *Elevarlo al cuadrado, obteniéndose 8 cifras. Si es necesario, añadir ceros por la izquierda hasta las 8.*

3.- *Tomar las 4 cifras centrales como el siguiente número Z_1.*

4.- *Poner la coma decimal a la izquierda para obtener un número entre (0,1).*

5.- *Tomar Z_1 como la semilla del siguiente.*

Respecto al algoritmo anterior, cabe realizar dos puntualizaciones: el paso 4 se hace porque normalmente en aplicaciones matemáticas se requieren números aleatorios reales entre 0 y 1, ya que mediante una simple multiplicación se podría obtener un conjunto de números entre cualquier rango entero. Si la aplicación requiere números enteros en vez de reales, se puede suprimir el punto 4. El otro punto a resaltar es el uso en el paso 5 del concepto de semilla que se repetirá a lo largo del artículo y se corresponde con el primer número que usamos para obtener la serie. Para obtener la semilla inicial Z_0, puede ser útil el método anteriormente mencionado del reloj: se asigna como semilla el valor actual del reloj (en los programas en C suele aparecer como *srand((unsigned) time(&t));* y en otros lenguajes *randomize(time);).*

En la figura 1 aparece un ejemplo con una serie de números, generados con este algoritmo.

$$Z_i = (aZ_{i-1} + c) \bmod M$$

donde
a: multiplicador
c: incremento o desplazamiento
M: módulo

Figura 2. Fórmula recursiva de los generadores congruenciales lineales.

i	Zi
0	7182
1	9921
2	13760
3	59
4	12834
5	7973

Figura 3. Ejemplo de aplicación de un generador congruencial lineal con a=133,c=4987 y m=16384

Posteriormente se comprobó que este método tiene el problema de que tiende a degenerar rápidamente a 0, momento en el cual repite indefinidamente ese valor (luego se verá que este problema también aparece en otros algoritmos, en los cuales si la semilla toma el valor 0 todos los números generados a continuación serán 0). Otro problema que presenta este método es que puede entrar en un ciclo corto, en el cual se limita a generar una y otra vez la misma secuencia de números. Una solución a esto podría ser cambiar la semilla cada cierta cantidad de números generados.

Con el ejemplo anterior, se ha visto que no todos los generadores son buenos. Hay ciertas propiedades que se le deben pedir a un buen generador:

1.- Sobre todo, los números producidos deben estar uniformemente distri-

Figura 4(a). Distribución 2D de un generador lineal, tomando la parte inferior del número generado.

buidos en el rango en el que se mueva y no debe aparecer correlación entre ellos (las propiedades enumeradas anteriormente).

2.- Desde un punto de vista práctico, debemos pedir que el generador sea rápido. La alternativa a esto sería mantener en memoria una tabla de números ya generados, lo que sería muy costoso en espacio.

3.- Debemos poder ser capaces de reproducir exactamente una secuencia dada de números aleatorios, para:

- hacer más fácil la depuración y la verificación del programa
- poder comparar dos sistemas distintos con los mismos números aleatorios, para comprobar qué sistema responde mejor

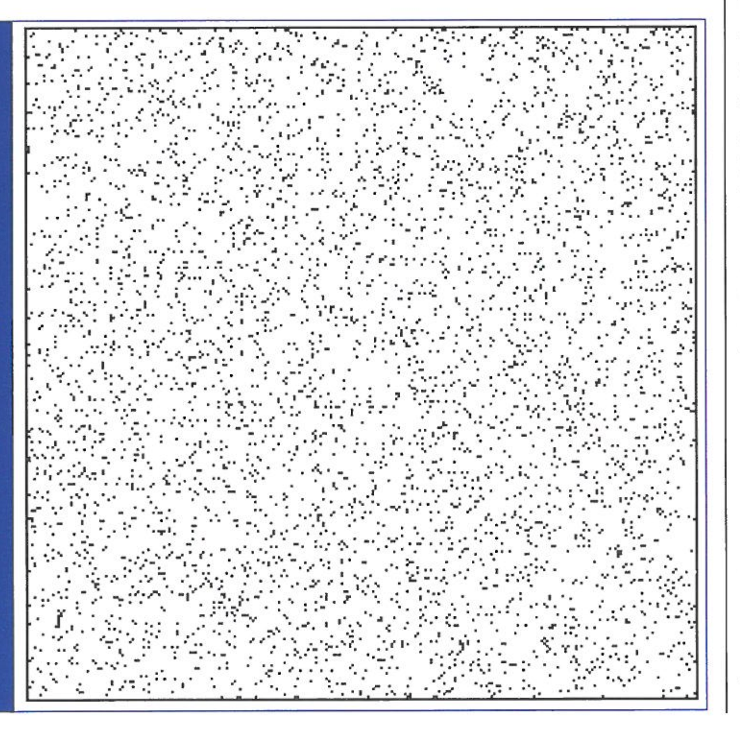


Figura 4(b). Distribución 2D de un generador lineal, tomando la parte superior del número generado.

A continuación se presentan los grupos de métodos que actualmente se usan en la generación de números aleatorios.

GENERADORES CONGRUENCIALES

Son los más utilizados (con ellos se implementa la función *random* de C). Siguen la fórmula recursiva que aparece en la figura 2.

Si el desplazamiento *c* es 0, el generador se denomina congruencial multiplicativo, que fue descrito en 1951 por Lehme. Si c es distinto de 0, el generador es congruencial lineal y fue presentado por Rotenburg en 1960. Como se puede apreciar cada número generado depende del anterior, a excepción de la semilla inicial Z0. En la figura 3 se presenta un ejemplo con varios pasos, tomando a=133, c=4987 y M=16384.

Todos los valores que aparecen son positivos y se suelen seguir las siguientes relaciones a la hora de elegir a, c, Z0 y M:

$0<M$, $a<M$, $c<M$ y $Z0<M$.

Por supuesto, no se puede decir estrictamente que los números generados sean aleatorios ya que no cumplen una propiedad de éstos: la independencia entre los números generados ya que está perfectamente definido, dado un número, cuál es el siguiente que se generará. Por esto normalmente a los números generados por estos algoritmos se les llama pseudoaleatorios. Este último comentario se convierte en el principal problema que presentan los generadores congruenciales: cada vez que se repite la semilla, el ciclo se repite de nuevo. Por ejemplo, si se ha definido una semilla Z_0 y después de generar 100 números vuelve a aparecer ese número, puesto que se tomará como la siguiente semilla, se volverán a repetir exactamente esos 100 números y de nuevo la misma semilla indefinidamente ya que la única variable en la función es el número generado anteriormente. Este ciclo aparece en todos los generadores congruenciales ya que se trabaja en un rango finito de números (entre 0

y M_{-1}, por la definición de módulo), por lo tanto la máxima longitud de un ciclo puede ser M. A la longitud de un ciclo se le llama periodo y depende de la elección de a, c, M y Z_0. En el caso de que el periodo sea igual al módulo, el generador se llama de periodo completo. A continuación se presentan una reglas para la elección de estos parámetros para conseguir buenos generadores:

- M debe ser grande
- en aras de la eficiencia, es deseable evitar la división que envuelve calcular un módulo. Para ello se suele elegir M=2b (por ejemplo, b=31 o b=15), con lo que el módulo simplemente es una operación AND con (M-1), o aprovechando el desbordamiento entero, aparece directamente como resultado, teniendo cuidado con el bit de signo -si M=2b, c es impar y a-1 es divisible por 4, el generador será de periodo completo independientemente de Z0

El parámetro c se usa simplemente para desplazar los puntos por lo que no interviene en el nivel de correlación de los puntos. Se ha comprobado que no hay ventajas aparentes para otro valor que no sea 1.

El caso de los generadores multiplicativos (c=0) tiene alguna diferencia con los anteriores. Por ejemplo, estos generadores no pueden tener periodo completo, ya que si Z_{i-1} tomase el valor de 0 ya no se generarían más números distintos de 0. Se pueden obtener generadores con periodo M_{-1} si se eligen bien M y a.

En este caso, si se elige M=2b para evitar la división, se ha demostrado que el periodo es como mucho 2b-2, es decir, M/4. Esto se consigue si Z0 es impar y a=8k+5 para k=0, 1,... Para alargar el periodo en vez de elegir M=2b se suele elegir el mayor primo menor que 2b.

Aparte de ser menor el rango estos generadores multiplicativos tienen otro problema y es que, si vemos en una recta los números generados, quedan muchos huecos lo que equivale a decir que no distribuye bien los valores.

Fórmula General:
$Z_i = g(Z_{i-1}, Z_{i-2},...) \bmod M$

Ejemplo A:
$g(Z_{i-1}, Z_{i-2},...) = a'Z^2{}_{i-1} + aZ_{i-2} + c$

Ejemplo B:
$g(Z_{i-1}, Z_{i-2},...) = a\ Z_{i-1}, Z_{i-2},...) = a_1 Z_{i-1} + a_2 Z_{1-2} + ... + a_q Z_{i-q}$

Figura 5. Fórmula general y dos ejemplos de un generador congruencial con más de una semilla.

$b_i = (c_i b_{i-1} + c_2 b_{i-2} + ... + c_q b_{i-q}) \bmod 2$

donde c_i = '{ 0, 1}

Ejemplo:
$b_i = (b_{i-r} + b_{i-q}) \bmod 2$

Figura 6. Fórmula general de los generadores de desplazamiento y ejemplo para 2 factores.

Respecto a los generadores aleatorios, hay que tener en cuenta que sólo podemos considerar realmente aleatorios al primer ciclo de números generados, ya que con la repetición de secuencias se pierde la imprevisibilidad. Así, si un generador tiene como longitud de su ciclo 1000 números no deberíamos usar más de esos 1000 números.

Una variación de los generadores lineales, que se usa en la implementación de 'rand' de C, consiste en trabajar siempre con los números más grandes que pueda manejar la máquina (por ejemplo, el tipo long de C, que usa 32 bits), pero quedarse sólo con una porción del número generado, normalmente un entero de 16 bits, con lo que se alarga el periodo. Además no da lo mismo elegir una porción que otra.

En la figura 4 se puede apreciar la diferencia de escoger la palabra menos significativa a la más significativa en un generador congruencial lineal según éste método, para 5000 números, con a=0x015a4e35, c=1 y M=32767.

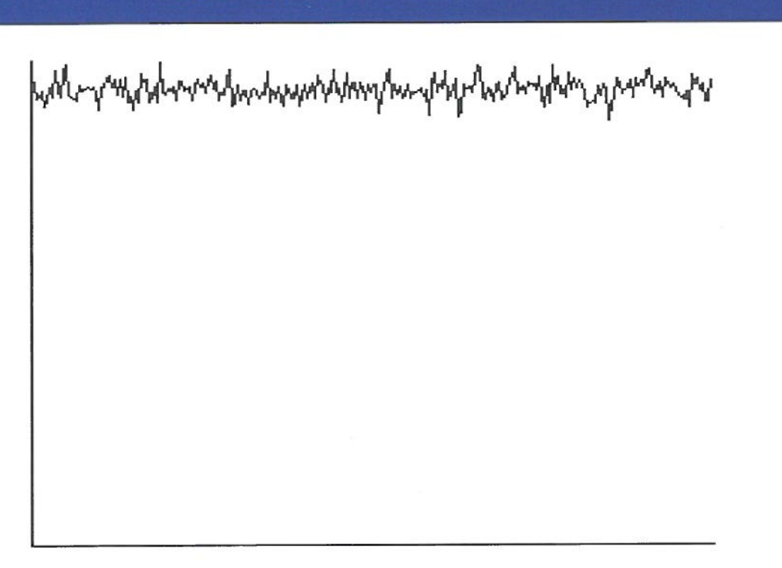
Además de los generadores multiplicativos y lineales, existen otros tipos de generadores congruenciales. En la figura 5 se puede apreciar la fórmula general y algunos ejemplos de un generador de números aleatorios que tiene en cuenta más de un factor ya generado.

La fórmula general simplemente indica que usamos una función que depende de varios números ya generados. En el ejemplo A se puede ver un generador congruencial cuadrático, donde a', a y c, son constantes. En el ejemplo B, los ai son también constantes. Con esta fórmula se pueden conseguir periodos gigantescos (M_{q-1}).

Del mismo modo también se pueden componer generadores, es decir, usar primero un generador de cualquier tipo y la salida de éste alimenta a otro generador. Se ha demostrado que la composición de generadores está mas cerca de la uniformidad real que cualquiera de sus componentes. Normalmente se usan varios generadores congruenciales lineales. Un modo de realizar esta composición es rellenar un vector secuencialmente con los primeros k números del primer generador (inicialmente se sugirió k=128, posteriormente se observó que con k=2 era suficiente). El segundo generador escoge un número entre 1 y k y se devuelve el número que ocupa esa posición en el vector antes preparado. El primer generador reemplaza la posición i con su muestra siguiente. El segundo generador elige otra posición y se repite el ciclo. Como se podrá apreciar estamos 'barajando' los números. Con este sistema dos generadores malos pueden formar otro bueno. Variaciones a este modelo han sido, por ejemplo, generadores que se barajan a sí mismos o aquéllos que realizan una permutación fija (por ejemplo, para k=4, seguir la secuencia 3, 1, 4 y 2 y volver a generar las 4 muestras, etc.). A pesar de las aparentes ventajas de la reordenación, no se sabe mucho de esta variedad de generadores.

Otra manera de combinar generadores es obtener dos Z_{1i} y Z_{2i} a partir de dos generadores con distintos módulos


Figura 7.- Histograma de un generador uniforme.



y devolver su resta módulo M. Las ventajas de estos generadores son que se consiguen periodos muy largos y que los multiplicadores de cada generador pueden ser pequeños.

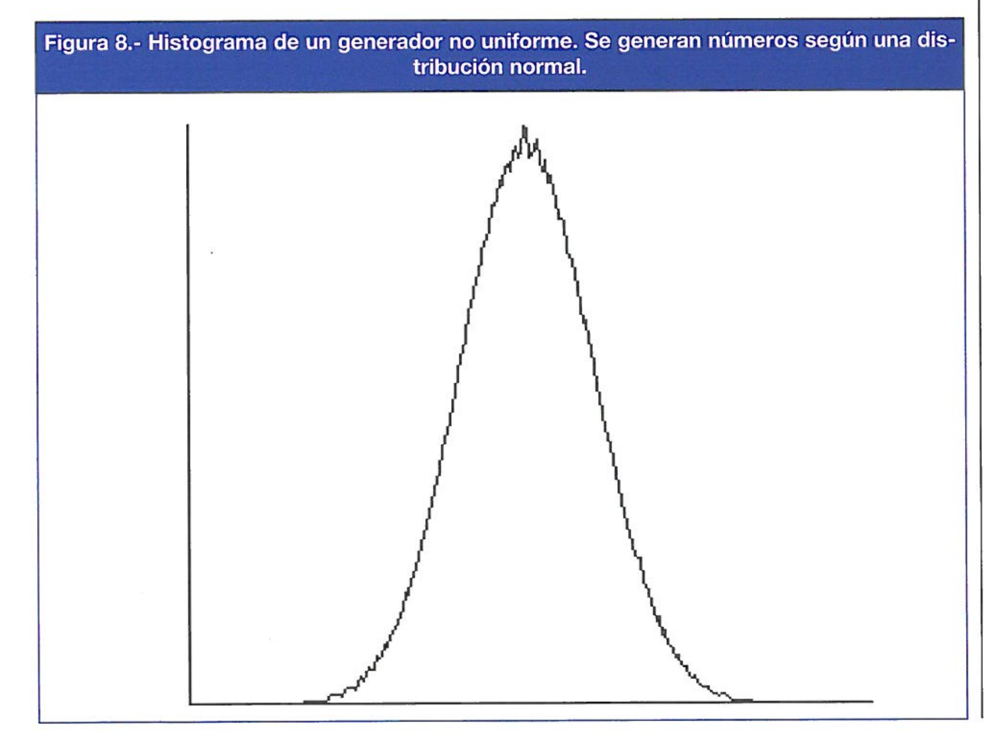
Por lo anteriormente visto, las posibilidades de la composición de generadores son inmensas, produciendo buenos resultados con operaciones simples.

Generadores de Registro de Desplazamiento.

Operan directamente con cadenas de bits. Dada una secuencia dada de bits b1b2b3b4..., se define la operación recursiva que aparece en la figura 6.

Como se puede apreciar, se trabaja a nivel de bit, donde cada uno depende de los q anteriores. El periodo máximo de estos generadores es 2_{q-1}. Normalmente sólo dos de los ci son distintos de 0, por lo que normalmente el generador queda como se muestra en el ejemplo de la figura 5, para enteros r y q tales que $0<r<q$.

Figura 8.- Histograma de un generador no uniforme. Se generan números según una distribución normal.



La operación anterior se puede calcular eficientemente mediante la operación lógica XOR, del siguiente modo: bi será 0 si bi-r y bi-q son iguales, y será 1 en otro caso. Para iniciar la secuencia se deben suministrar los primeros q dígitos (la semilla). Con estos generadores se pueden conseguir periodos increíbles (10156 o más).

GENERADORES NO UNIFORMES

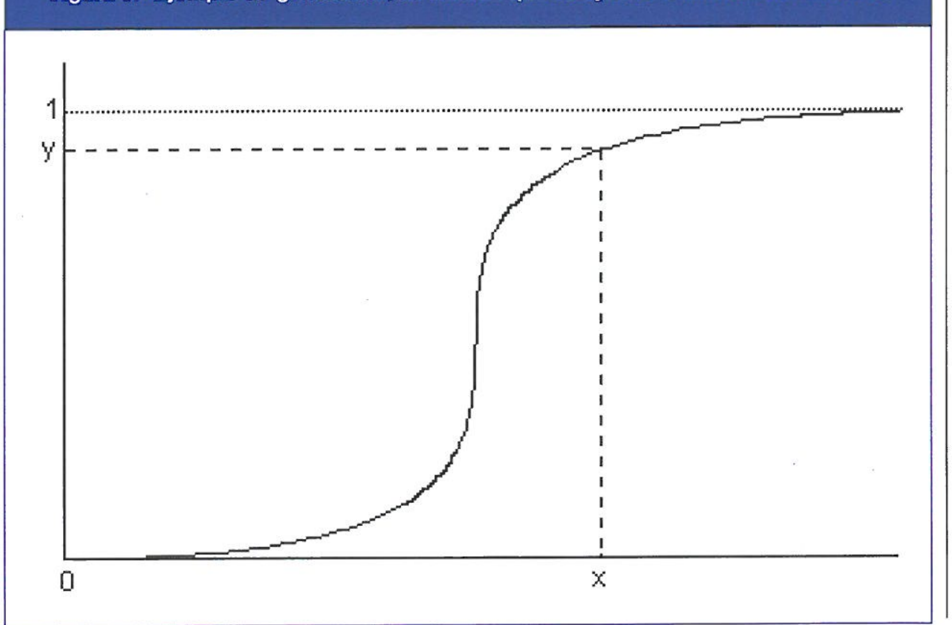
Hasta ahora se han visto generadores aleatorios uniformes, es decir, que cada valor que producen tiene la misma probabilidad. En este apartado se introducen otro tipo de generadores, muy importantes en campos de simulación y generación de test: los generadores no uniformes. Estos generadores concentran los valores de salida en uno o varios puntos y sus alrededores. Si se obtiene un histograma (distribución relativa de los distintos valores generados) a partir de una muestra suficientemente grande de un generador uniforme, se obtendrá una gráfica parecida a la que se muestra en la figura 7. En cambio, en la figura 8 se muestra el histograma de una serie de números obtenidos por un generador no uniforme que, como se estudia en cualquier curso de estadística, se ajusta a una distribución normal. Estos histogramas también se les llama funciones de densidad de una variable estadística.

Para implementar este tipo de generadores no uniformes se suelen usar dos métodos, que se presentan a continuación.

El primer método parte de la función de distribución de la variable aleatoria a la que se quiere que se ajusten los puntos generados. La función de distribución está definida de 0 a infinito y toma valores entre 0 y 1. El procedimiento consiste en generar valores uniformemente distribuidos entre 0 y 1 y calcular la inversa de esta función en el punto generado. Con esto obtenemos valores mayores de 0. En la figura 9 se puede ver un ejemplo para un generador normal.

El procedimiento a seguir en la generación de estos valores es sencillo:

Figura 9.- Ejemplo de generación de valores que se ajustan a una función normal.



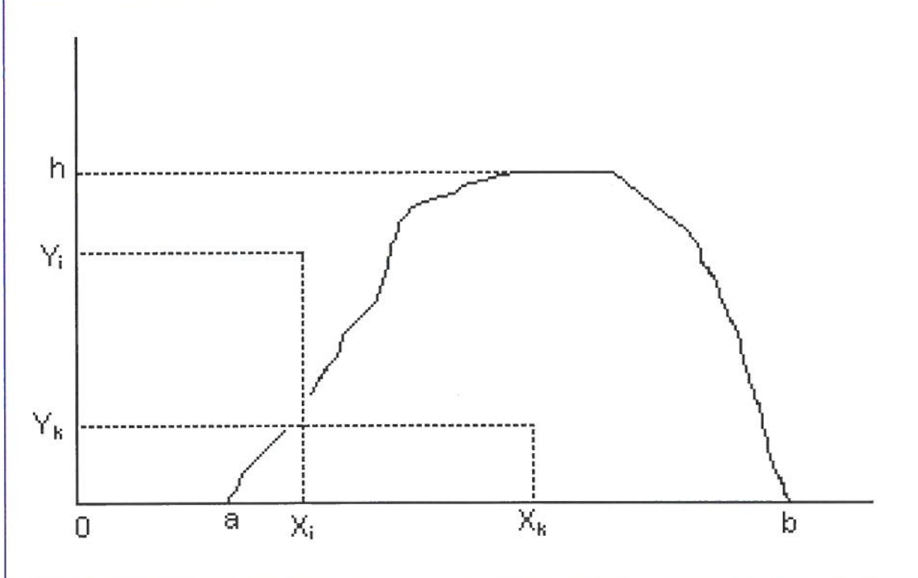
generar un valor 'y' uniformemente distribuido entre 0 y 1 con cualquiera de los métodos que se han visto, y se calcula la 'x', mediante la inversa de la función de distribución.

El segundo método es más sencillo y permite generar cualquier tipo de distribución, incluso aquellas que sería muy difícil expresar mediante una fórmula matemática. Se le conoce como el método de rechazo y requiere conocer el histograma de la distribución a generar, es decir, para cada valor a generar su probabilidad de aparición. A continuación, para cada punto a generar, se requieren dos valores, uno distribuido uniformemente entre 0 y 1 y el otro entre el conjunto de posibles valores que pueden aparecer. Uno representa la altura y otro representa el valor en el eje de abcisas. Si ese punto está por debajo de la curva de densidad entonces se acepta el valor de la abcisa como generado. Si cae fuera de la curva, se rechaza (de ahí le viene el nombre). En la figura 10 aparece un ejemplo donde, como se puede apreciar, se generaría el valor Xk y se rechazaría Xi.

En forma de pseudocódigo, el algoritmo del método de rechazo se detalla a continuación:

1.- Generar r1 y r2, números generados uniformemente entre 0 y 1.

Figura 10.- Ejemplo del método de rechazo. Se genera el valor Xk.



2.- Calcular:
Xi=a+(b-a)r1
Yi=hr2

3.- Si Yi>f(Xi), volver al paso 1, si no, generar Xi.

El valor de f(Xi) representa la altura del histograma en el punto de abcisa Xi. Este valor puede venir dado por una función matemática, o bien en un array en el que, para cada punto, se mantiene su altura correspondiente; de este modo se puede reproducir cualquier distribución.

RESUMIENDO

Como se puede apreciar, la generación de números aleatorios es, además de interesante, un tema muy denso. Este artículo no ha querido ser más que una introducción más o menos práctica al mundo de la generación de números aleatorios que, en contraposición con otros apartados de las matemáticas, es un tema vivo, en el que se están centrando muchos grupos de trabajo en la actualidad.

Para no quedar sólo en el papel, se ha incluido un programa de prueba de distintos generadores de números aleatorios. Es un programa para Windows en el que se puede seleccionar el tipo de generador a utilizar, además de sus parámetros (semilla, multiplicador, etc.). Para comprobar la bondad o no de un generador, se ha introducido un modo 'histograma', que muestra la distribución para comprobar visualmente la uniformidad o no de los valores. También se ha introducido un modo '2d' en el que se puede apreciar la posible correlación entre los valores generados, apareciendo esta correlación como líneas o curvas entre los puntos. Con este programa se puede apreciar la influencia que tiene, por ejemplo, la elección del multiplicador en un generador lineal. Junto al programa se adjunta un archivo con las instrucciones de uso.

CREACIÓN DE COMPONENTES

Enrique de la Lastra

El lenguaje de alto nivel en el que se programa en Delphi es Object Pascal, una ampliación del Turbo Pascal, a la que se ha añadido la programación orientada a objetos (OOP: *Object Oriented Programing*). El estilo de programación orientado a objetos es el más adecuado en entornos gráficos ya que se encapsula toda la funcionalidad y todas las propiedades de cada objeto dentro del mismo. Antes de explicar qué es un componente es conveniente aclarar algunos conceptos que provienen, en su mayoría, de este método de programación y que son útiles para entender lo que se va a explicar a lo largo de este artículo (y de sucesivos artículos que irán apareciendo en esta revista) así como para familiarizarse con la terminología utilizada en Delphi.

Lo primero es definir un objeto. Una definición más o menos genérica es que un **objeto** es la representación informática de un objeto físico o de un concepto. En realidad dicha representación informática se adapta mejor a lo que se denomina **clase** y el objeto no es más que una instancia de una clase. Esto se puede entender mejor con un ejemplo: si definimos un HOMBRE con una serie de atributos como

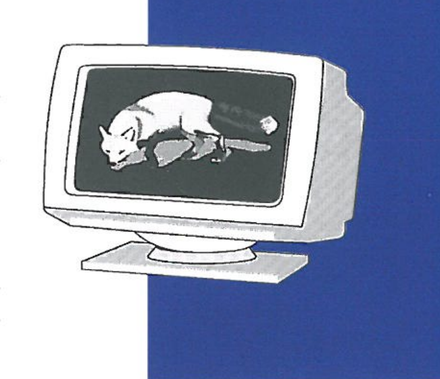
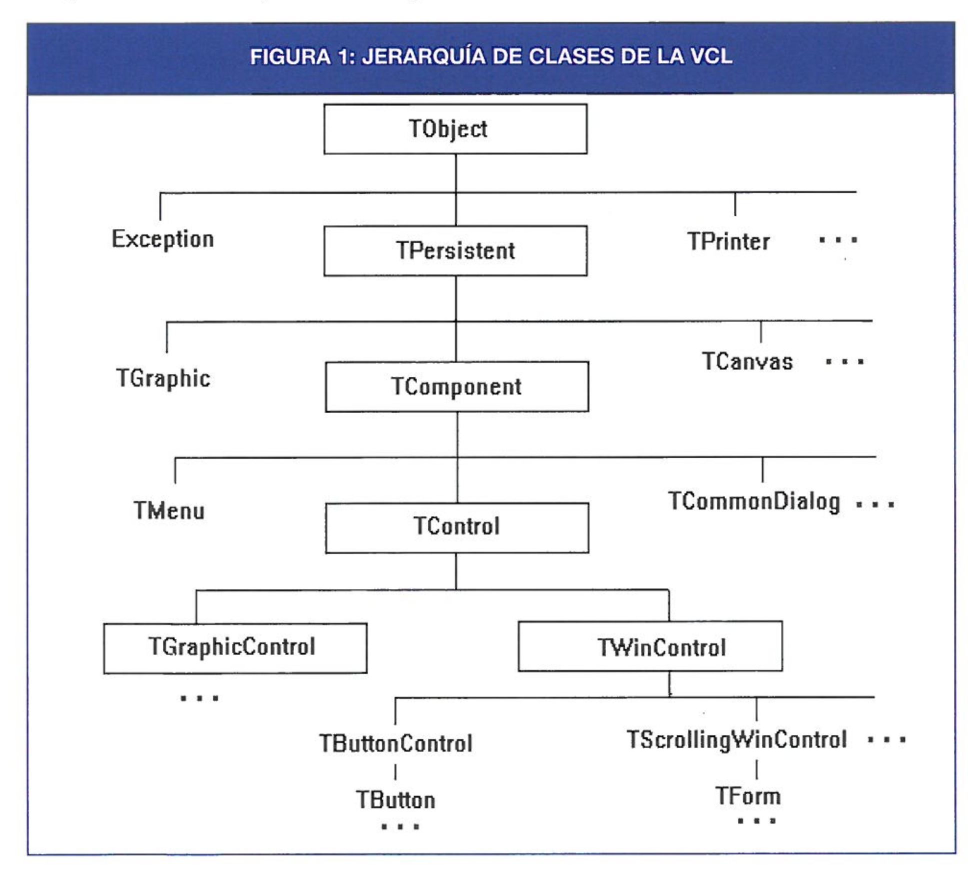


FIGURA 1: JERARQUÍA DE CLASES DE LA VCL



El Compilador Delphi es una poderosa herramienta de desarrollo software que permite el diseño y la creación visual de aplicaciones en el entorno windows. El éxito alcanzado por Delphi en el mundo de la programación se debe a que combina en un solo producto potencia y facilidad de desarrollo.

son sexo, edad, altura, peso, etc., la "clase" será la propia definición de un hombre mientras que objetos serán cada uno de los individuos "humanos" que resulten de particularizar las características de dicha clase para cada uno de ellos. Esto es lo que significa que un objeto sea una "instancia" de una clase así como "instanciación" es la particularización de una clase en un objeto.

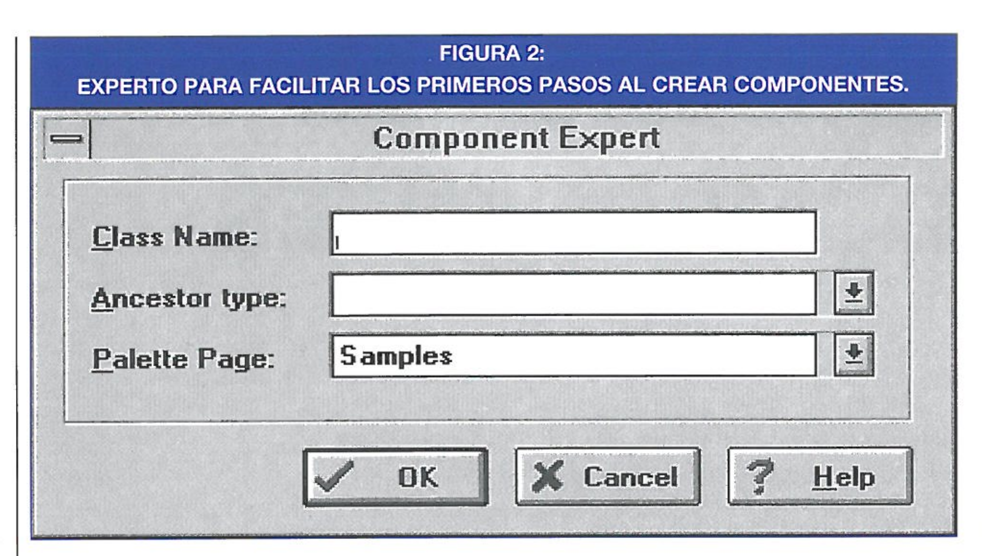
Y entonces ¿qué es un componente?. Un **componente** es un **objeto** al que se añade la característica de ser reutilizable de diferentes maneras para diferentes aplicaciones. Los componentes son los elementos fundamentales con los que se realizan aplicaciones en Delphi. Se puede ver un componente como un hijo que hereda de su padre (un objeto) todas las características de éste y le añade otras nuevas. Y ésta es la definición de **herencia:** la capacidad de aprovechar las características de una clase ya definida (por ejemplo una clase SER_VIVO) y añadir otras nuevas (por ejemplo una clase SER_HUMANO, hija de SER_VIVO). En este punto, hay que destacar que Delphi sólo soporta la "herencia simple", es decir la herencia de una sola clase y no la de varias clases a la vez ("herencia múltiple") que sí soportan otros lenguajes como por ejemplo C++.

Quedan por definir dos conceptos importantes de la OOP. El primero es la **encapsulación,** que se puede definir como la posibilidad de esconder las características de implementación del objeto y mostrar sólo al usuario del propio objeto la interfaz desde la cual acceder al servicio ofrecido por el mismo (ver siguiente apartado para entender un poco mejor este término). El otro concepto es el de **polimorfismo** que, sin llegar a una definición académica, es lo que permite a una clase redefinir métodos de la clase padre para adaptarlos a los requisitos de aquélla. Para que esto sea posible los métodos de la clase padre han de ser *virtual* (virtuales), como se explicará después.

OTROS TÉRMINOS IMPORTANTES

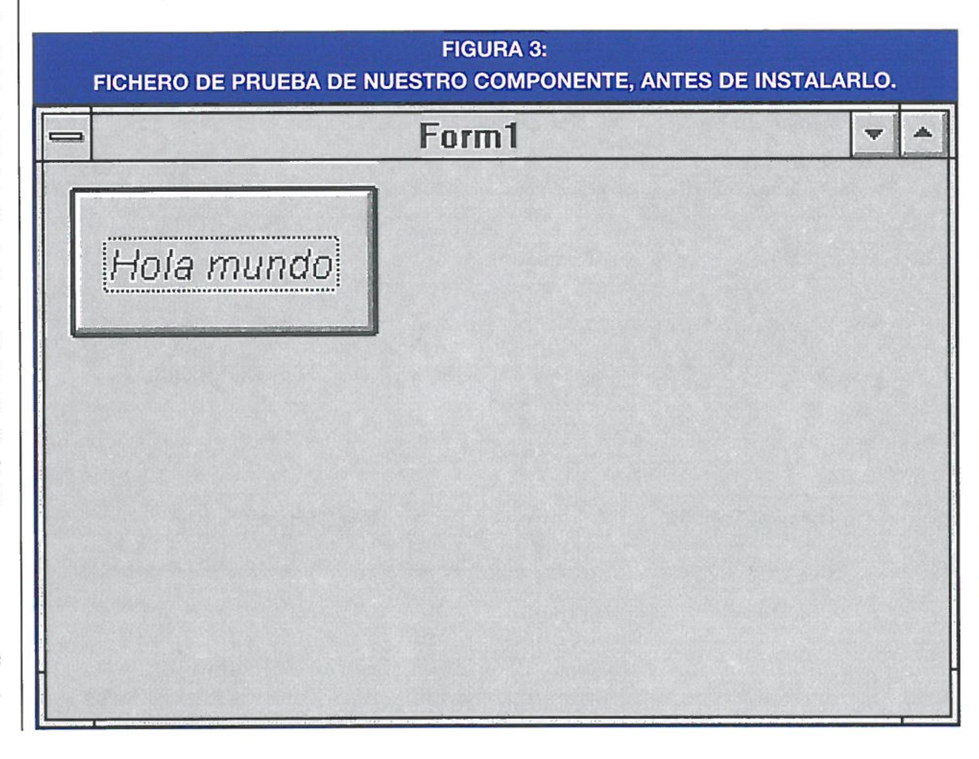
Hay otros conceptos imprescindibles de conocer en las clases de Delphi. Pasamos a definirlos:

FIGURA 2: EXPERTO PARA FACILITAR LOS PRIMEROS PASOS AL CREAR COMPONENTES.



- **Niveles de ocultación:** existen varias palabras reservadas utilizadas para proveer la necesaria ocultación de datos. Básicamente hay dos niveles de ocultación: la parte privada y la parte pública de la clase.
  - **private:** parte de una clase que sólo es accesible desde los objetos (instancias) de la propia clase.
  - **protected:** igual que la anterior pero en este caso también pueden acceder a esta parte los objetos de las clases hijas.
  - **public:** parte accesible por todos los objetos.
  - **published:** igual que la anterior pero en este caso esta parte es accesible también por el *Object Inspector* (inspector de objetos de Delphi).
- **Variables**: campos de datos públicos o privados.
- **Propiedades**: atributos de la clase que definen las acciones a tomar cuando se lee o escribe dicho atributo. Generalmente serán *public* o *published* (por ejemplo: la propiedad *Caption* de un botón).
- **Eventos**: sucesos ocurridos en un objeto. Desde un punto de vista téc-

FIGURA 3: FICHERO DE PRUEBA DE NUESTRO COMPONENTE, ANTES DE INSTALARLO.



LISTADO 1:
CÓDIGO CREADO POR EL COMPONENT EXPERT.

C:\DELPHI\BIN\UNIT1.PAS

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls;

type
  TNuestroBtn = class(TButton)
  private
    { Private declarations }
  protected
    { Protected declarations }
  public
    { Public declarations }
  published
    { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Standard', [TNuestroBtn]);
end;

end.
```

30: 5 Modified Insert

Unit1

nico un evento es el modo de suministrar un enlace entre windows y el objeto, de forma que éste pueda manejar lo que ocurre tras un suceso en el que "se ve implicado". En próximos artículos se explicarán en profundidad los eventos, su relación con los mensajes de windows y cómo tratar dichos eventos.

Una vez instalado un componente nuevo podremos utilizarlo como cualquier otro de la paleta

- **Procedimientos**: son los elementos de la clase que permiten "hacer algo". Pueden devolver un valor (ser *function*, funciones) o no hacerlo (ser *procedure*, procedimientos).

- **Métodos**: o "manejadores de eventos" son los procedimientos que proporcionan una funcionalidad ante un evento concreto.

  - **métodos virtual:** métodos definidos en una clase padre que pueden ser redefinidos en la clase hija para añadir alguna funcionalidad.

  - **métodos static:** el resto de los métodos, que si se redefinen, lo hacen completamente sin dejar opción de acceso a lo definido en la clase padre.

- **Constructor y destructor:**

  - **constructor:** es un método que se utiliza para crear e inicializar las variables y propiedades de un objeto.

  - **destructor:** es un método, llamado automáticamente cuando el objeto es destruido por Windows, que sirve para destruir aquello para lo que hemos asignado memoria.

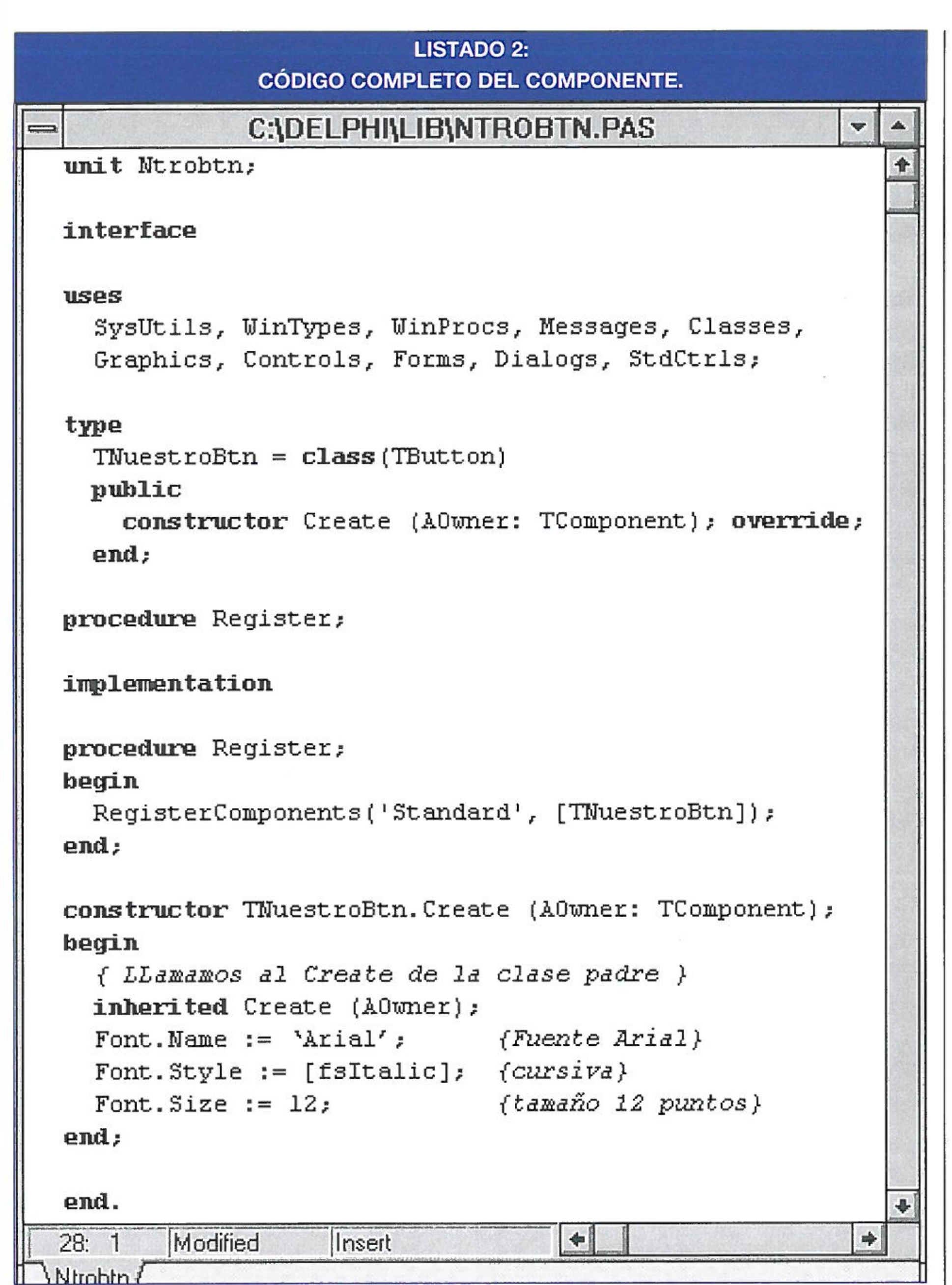
JERARQUÍA DE CLASES EN DELPHI

En el entorno de desarrollo Borland Delphi, todas las clases heredan en primera instancia de la clase TObject. A los objetos que heredan directamente de Tobject sólo podemos acceder en tiempo de ejecución (es decir siempre que la aplicación esté ejecutándose) y nunca en tiempo de diseño (visualmente, antes de compilar y ejecutar la aplicación).

La librería de clases de Delphi, la llamada VCL o Visual Component Library (Biblioteca de Componentes Visuales), tiene como padre jerárquico de todas sus clases a la clase TComponent, que hereda directamente de TObject. De ahí que las clases de la VCL se denominen componentes. Los componentes son los bloques con los que se construyen las aplicaciones de Delphi. Tienen propiedades, métodos y eventos que los distinguen del resto. A diferencia de los objetos (que heredan de Tobject), se puede acceder a los componentes tanto en tiempo de diseño como en tiempo de ejecución. Un tipo especial de componentes son los denominados controles.

Los controles son componentes visuales, es decir, son visibles cuando

LISTADO 2:
CÓDIGO COMPLETO DEL COMPONENTE.


C:\DELPHI\LIB\NTROBTN.PAS

```
unit Ntrobtn;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls;

type
  TNuestroBtn = class(TButton)
  public
    constructor Create (AOwner: TComponent); override;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Standard', [TNuestroBtn]);
end;

constructor TNuestroBtn.Create (AOwner: TComponent);
begin
  { LLamamos al Create de la clase padre }
  inherited Create (AOwner);
  Font.Name := 'Arial';        {Fuente Arial}
  Font.Style := [fsItalic];    {cursiva}
  Font.Size := 12;             {tamaño 12 puntos}
end;

end.
```


la aplicación está ejecutándose (ejemplos claros en Windows son un botón, una casilla de marca o una caja de texto). Esta característica los diferencia del resto de los componentes, que no son visibles en tiempo de ejecución. Todos los controles heredan de TControl que, a su vez, hereda de TComponent.

Cualquier clase que se crea en Delphi hereda directa o indirectamente de Tobject, aunque lo normal es heredar en algún nivel mayor de la jerarquía (es decir de Tcomponent, TControl o cualquier clase hija de ellas).

La jerarquía de clases que sigue la VCL se resume en el esquema de la figura 1.

Delphi permite no sólo usar los componentes de la VCL sino además crear nuevos componentes que se adapten a las necesidades del programador. Crear un nuevo componente puede ser tan simple como añadir una propiedad a un componente ya existente o puede requerir una gran labor de programación en los casos en los que el programador deba desarrollar un componente enteramente.

La posibilidad de creación de componentes dentro del propio entorno de desarrollo supone actualmente una gran ventaja frente a productos de la competencia en los que para la creación de controles hay que cambiar de herramienta y hasta de lenguaje de programación.

CREAR UN COMPONENTE

Al implementar un nuevo componente se puede optar por distintas alternativas:

- Cambiar las propiedades por defecto de un componente existente.
- Añadir propiedades y funcionalidades nuevas a un componente.
- Crear completamente un componente a medida.

Por supuesto, y gracias a la programación orientada a objetos, podremos crear un nuevo componente basado en uno que nosotros mismos hayamos creado, al que queremos añadir una funcionalidad distinta o mejorada. De hecho no estamos haciendo más que extender la jerarquía de clases en uno o más niveles.

Una vez creado e instalado un componente, podremos utilizarlo como cualquier otro de la VCL, es decir, seleccionándolo de la paleta de componentes y haciendo click en un formulario. Todas las propiedades y eventos nuevos que hayamos definido como published aparecerán en el Object Inspector y podremos modificarlas tanto en tiempo de diseño como en tiempo de ejecución.

NUESTRO PRIMER COMPONENTE

Para crear nuestro primer componente elegimos la opción más sencilla, que será la de modificar las propiedades por defecto de un componente de la VCL. En posteriores artículos se explicará cómo desarrollar componentes más complejos en los que definiremos nuevos eventos y propiedades para adaptarlos a cada necesidad concreta.

Elegimos un botón, de la clase TButton cuya propiedad Font (fuente) es por defecto de tipo System, con estilo normal (no es negrita, ni cursiva) y de tamaño 10 puntos. Supongamos que para una aplicación deseásemos que todos los botones tuviesen como fuente Arial, con estilo en cursiva y fuente 12 puntos. Tendríamos dos opciones; la

LISTADO 3:
CÓDIGO DEL MÓDULO DE PRUEBA DEL COMPONENTE.

UNIT1.PAS

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, Forms, Dialogs, NtroBtn;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  public
    NuestroBtn: TNuestroBtn;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  NuestroBtn := TNuestroBtn.Create (Self);
  NuestroBtn.Parent := Self;
  NuestroBtn.Left := 10;
  NuestroBtn.Top := 10;
  NuestroBtn.Width := 120;
  NuestroBtn.Height := 60;
  NuestroBtn.Caption := 'Hola mundo';
end;

end.
```

12: 13 | Insert

Unit1

primera, que cada vez que se añadiese un botón, editásemos desde el Object Inspector dicha propiedad y cambiásemos uno a uno dichos atributos; la segunda, que es la que nos interesa, que al seleccionar un botón tuviese como fuente por defecto la deseada. Para hacer esto último, necesitamos definir un nuevo componente. A la clase de este componente la denominaremos: TNuestroBtn.

Los pasos a seguir para crear un nuevo componente Delphi son:

1. Heredar de un componente de la VCL (si no se elige explícitamente, todos los objetos heredan implícitamente de TObject). En nuestro caso, el componente heredará de la clase TButton para lo cual es preciso escribir lo siguiente en la parte interface del fichero que va a contener el código del componente:

```
type TnuestroBtn = class (TButton)
```

2. Definir las variables como propiedades y eventos nuevos. Aquí no vamos a definir ninguna característica nueva, sólo cambiar una de las existentes. Veremos más adelante cómo se hace.

3. Registrar el componente. Este paso es necesario para informarle a Delphi en qué pestaña de la paleta de componentes queremos que aparezca. Para registrar un componente hay que añadir en la parte interface:

```
procedure Register;
```

Y en la parte implementation:

```
procedure Register;
begin
 RegisterComponents ('Standard', [TNuestroBtn]);
end;
```

Esto registrará el componente TNuestroBtn en la pestaña Standard de la paleta de componentes.

Para agilizar este proceso existe una utilidad en Delphi que ahorra al programador la necesidad de escribir las líneas anteriores. Se trata del Component Expert (experto para componentes). Para utilizarlo habrá que elegir la opción New Component... del menú File y aparecerá el diálogo de la figura 2.

En este diálogo habrá que escribir en Class name (nombre de la clase): TNuestroBtn. En Ancestor type (tipo de la clase padre): TButton. En Palette page (pestaña de la paleta de componentes): Standard. Después, pulsar el botón OK. Se creará una nueva unit con el código que aparece en el listado 1.

Ya sólo queda añadir el código que cambia la fuente por defecto. Este código debe ir en el constructor de la clase, para que el cambio de fuente se realice con sólo seleccionar el componente desde la paleta. El código que hay que añadir en la parte implementation es el siguiente:

```
constructor TNuestroBtn.Create (AOwner: TComponent);
begin
 inherited Create (AOwner);
 Font.Name := 'Arial';     {Fuente Arial}
 Font.Style := [fsItalic]; {cursiva}
 Font.Size := 12;          {tamaño 12 puntos}
end;
```

La línea: inherited Create (AOwner), llama al constructor de la clase padre (para que se hagan las inicializaciones necesarias). Pero, ¿cómo es posible que llamemos al Create de la clase padre si en nuestra clase

TNuestroBtn el constructor también tiene el nombre Create?. La razón es el polimorfismo y el que el método Create es virtual, lo cual permite redefinirlo sin perder el acceso al método de la clase padre. Sin embargo, para redefinir el constructor, hay que declararlo de la siguiente manera en la parte interface:

```
public
  constructor Create (AOwner: TComponent); override;
```

Debe ir en public para que sea posible crear los objetos instanciados desde cualquier otro objeto. Debe, además, llevar el parámetro override (no hacer caso), para que sea accesible el constructor de la clase padre y, a su vez, podamos redefinirlo. Si no pusiéramos override, podríamos redefinirlo pero, sin embargo, no tendríamos acceso al constructor del padre perdiendo de esta manera toda la inicialización que éste hace (acción muy peligrosa ya que el constructor del padre se encarga de toda la asignación de memoria necesaria para el objeto).

Y ya está. Ahora tenemos que salvar la unit con un nombre apropiado como por ejemplo, NtroBtn. Para ello, en el menú File elegimos la opción Save File As... y tecleamos: NtroBtn.pas.

Por fin, el código completo del componente que hemos creado quedaría como se muestra en el listado 2.

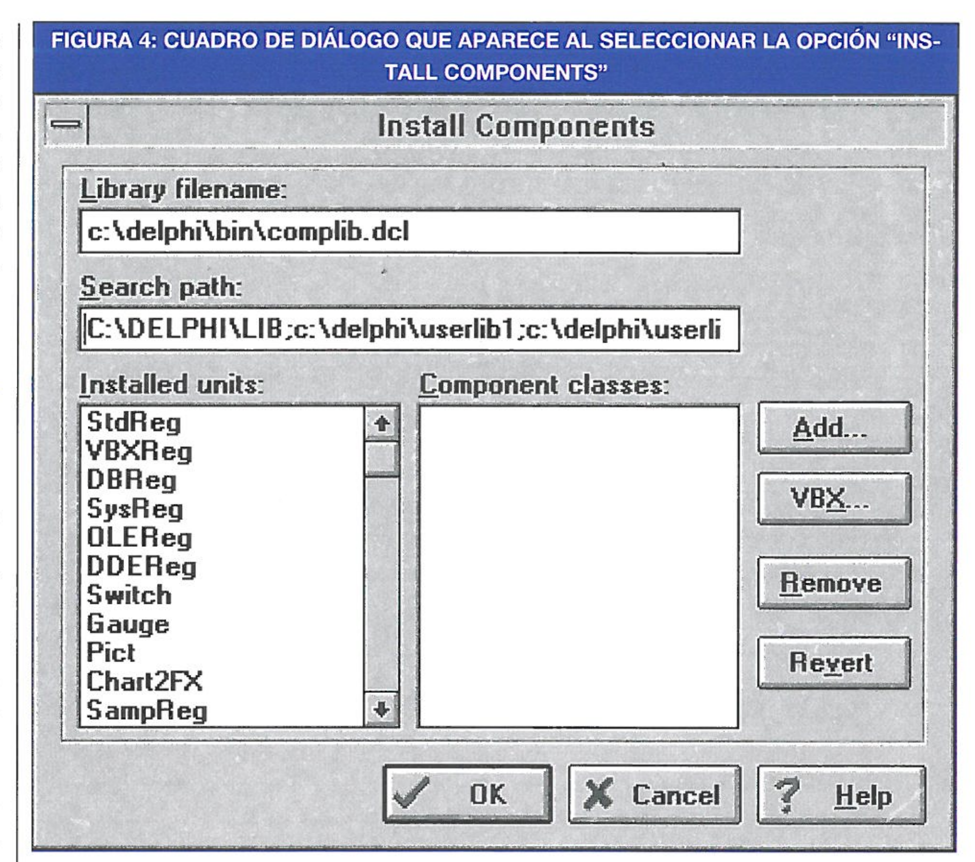
COMPROBAR EL COMPONENTE

Una vez que se ha creado un componente para que podamos usarlo como cualquier otro de la paleta de componentes hay que instalarlo. Instalar un componente supone añadir su código objeto al de la librería de componentes. Esta librería, que contiene todos los componentes que aparecen en la paleta de componentes, se almacena en el archivo COMPLIB.DCL. Cada vez que instalemos un componente nuevo añadiremos su código al de la librería, la cual irá "engordando" según vayamos incorporando nuevos componentes.

Antes de instalar el componente debemos comprobar que está escrito de forma correcta (al menos, sintácticamente). La forma más sencilla de realizarlo es seleccionar en el menú Run la opción Syntax Check, que realiza una comprobación sintáctica de una unit.

Sin embargo, lo más correcto es compilar el código incorporándolo a un proyecto y crear el componente manualmente. Este paso es aconsejable por dos razones. La primera, porque instalar el componente supone recompilar la librería, incluido el nuevo componente, con lo que nos ahorramos todos lo posibles errores que aparezcan debidos a un código mal escrito. la segunda, y quizá más importante, porque recompilar la librería lleva su tiempo y en máquinas "lentas" (familia 486) el proceso puede durar varios minutos.

FIGURA 4: CUADRO DE DIÁLOGO QUE APARECE AL SELECCIONAR LA OPCIÓN "INSTALL COMPONENTS"



Los pasos para compilar nuestro componente son los siguientes: abrir un nuevo proyecto (Project1); añadir en la parte uses de cualquier unit del mismo (unit1), el nombre del fichero que contiene nuestro componente (NtroBtn); incorporar en la parte public del formulario (TForm1)el nombre del componente:

```
public
  NuestroBtn: TNuestroBtn;
```

El último paso es seleccionar el método Create del formulario (FormCreate) y escribir dentro las siguientes líneas de código:

FIGURA 5: CUADRO DE DIÁLOGO PARA AÑADIR EL NOMBRE DEL FICHERO QUE CONTIENE EL COMPONENTE

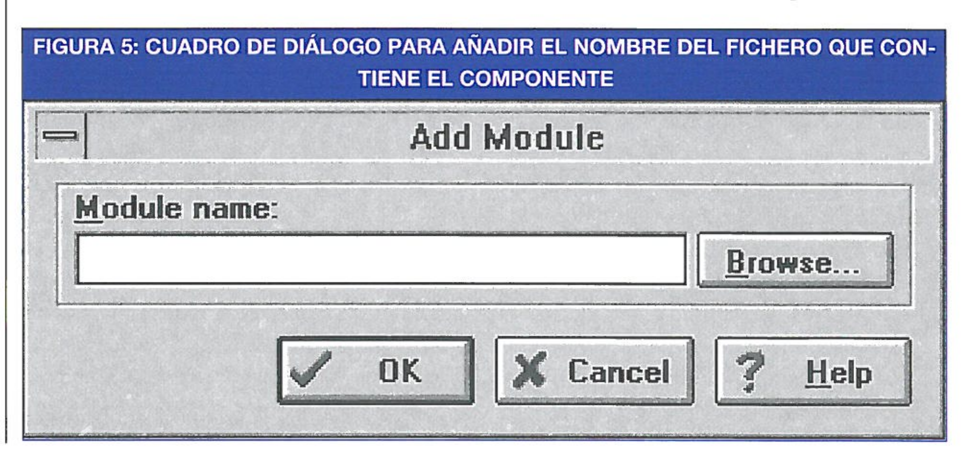
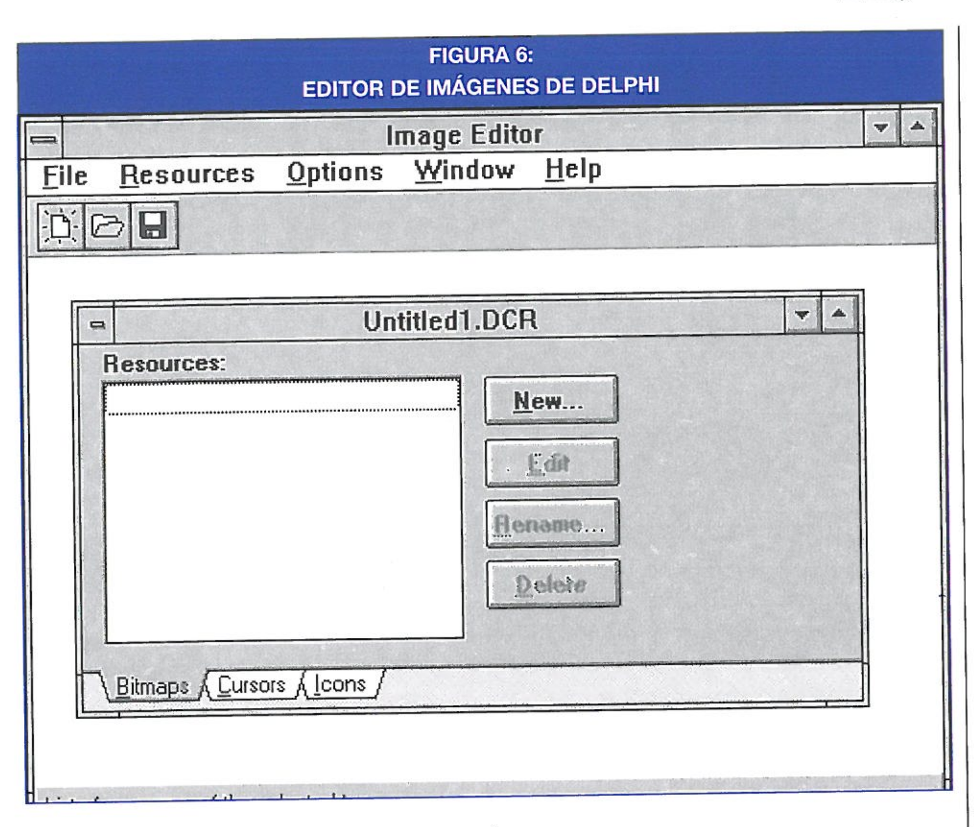


FIGURA 6:
EDITOR DE IMÁGENES DE DELPHI




```
procedure TForm1.FormCreate (Sender: TObject);
begin
  ...
  NuestroBtn := TNuestroBtn.Create (Self);
  NuestroBtn.Parent := Self;
  NuestroBtn.Left := 10;
  NuestroBtn.Top := 10;
  NuestroBtn.Width := 120;
  NuestroBtn.Height := 60;
  NuestroBtn.Caption := 'Hola mundo';
  ...
end;
```

En el Create pasamos como parámetro Self, lo que quiere decir "el propio formulario" (Form1), ya que el "dueño" de nuestro botón será el formulario. La línea: NuestroBtn.Parent:= Self asigna como padre del botón también al formulario (Form1). La diferencia entre el dueño y el padre se verá en otro artículo pero baste saber ahora que el dueño ha de ser el formulario, mientras que el padre puede ser el formulario, o un panel... En cuanto al resto de las líneas, sirven para situar manualmente el botón en el formulario con el tamaño deseado y con el título: "Hola mundo". Este código se muestra en el listado 3.

Al final, compilamos el proyecto y si todo está correcto, al ejecutar la aplicación veremos aparecer nuestro botón con las características asignadas por defecto y la posición elegida dentro del formulario (figura 3).

INSTALAR UN COMPONENTE

Sabiendo que el componente es sintácticamente correcto y que, además, funciona como deseábamos, estamos preparados para instalarlo de forma que se agregue a la paleta de componentes. Para ello, se selecciona en el menú Options la opción Install Components... y aparecerá la pantalla que se muestra en la figura 4.

Pulsamos el botón Add... y se nos muestra el diálogo de la figura 5.

En la caja de texto introducimos el nombre del fichero que contiene nuestro componente, con la ruta completa. Si no nos acordamos de la ruta, pulsamos el botón Browse y buscamos el archivo. Cabe mencionar aquí que podríamos seleccionar tanto el fichero con el código fuente (*.PAS) como el fichero ya compilado (*.DCU). Esta última opción permite instalar componentes de los que no tengamos el código fuente. Pulsamos el botón OK y Delphi comenzará a recompilar la librería de componentes. Este proceso puede llevar varios minutos en máquinas 486 y menos de un minuto en máquinas Pentium.

Al terminar, aparecerá un nuevo botón en la pestaña Standard de la paleta. A partir de este momento, podremos utilizarlo como cualquier otro de los ya existentes.

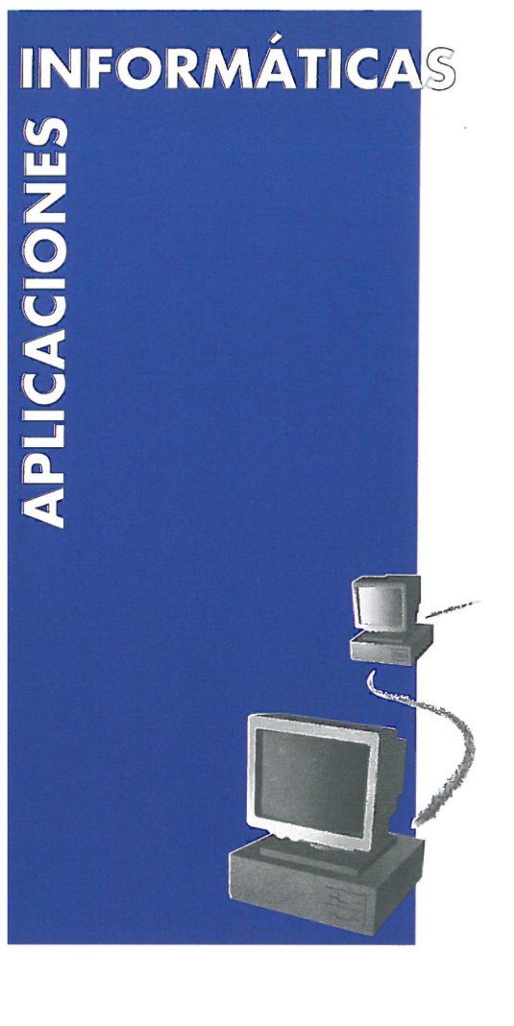
CAMBIAR EL BITMAP DEL COMPONENTE EN LA PALETA

Instalado ya el componente en la paleta, se observa que el bitmap con el que aparece es igual al de la clase padre (TButton). Es decir, es imposible saber a simple vista cuál es el TButton original y cuál nuestro TNuestroBtn ya que comparten apariencia en la paleta de componentes (aunque evidentemente, si dejamos el ratón quieto sobre cada uno, aparecerá el hint o aviso que informará del nombre de la clase).

Si queremos que el componente creado se muestre de una forma distinta, deberemos crear un archivo con extensión *.DCR. Para ello, utilizaremos el Image Editor (editor de imágenes) que viene suministrado con Delphi. Seleccionamos la opción Image Editor del menú Tools.

Dentro del editor de imágenes elegimos la opción New del menú File y en el diálogo que aparece, seleccionamos: Component Resource (DCR). Obtendremos la imagen de la figura 6.

Pulsamos el botón New... y en el diálogo siguiente elegimos Bitmap y pulsamos OK. El tamaño del bitmap ha de ser de 24X24 pixels. Hacemos click en OK y ya podemos dibujar el bitmap para nuestro botón. Una vez dibujado cambiamos el nombre de dicho bitmap con el botón Rename..., para que tenga el mismo nombre que el componente (TNUESTROBTN, en mayúsculas por convenio) y eligiendo la opción Save As... del menú File, guardamos el fichero en la misma ruta y con el mismo nombre que el archivo que contiene el componente pero con extensión *.DCR (en nuestro caso NtroBtn.DCR). Ahora habrá que reinstalar el componente para que enlace el fichero de recursos: NtroBtn.DCR, junto al fichero del componente: NtroBtn.PAS. A diferencia de como se hizo anteriormente, ahora basta con elegir la opción Rebuild Library (reconstruir la librería) del menú Options. Una vez terminada la reconstrucción de la librería el bitmap editado será el que aparezca en la paleta de componentes.



LA ARQUITECTURA CLIENTE/SERVIDOR

Miguel González Maestre

La última gran apuesta de importantes empresas informáticas son aplicaciones y sistemas basados en el modelo Cliente/Servidor. Intentaremos explicar el estado del arte y las posibles novedades en un sector que cambia vertiginosamente.

LA SEGUNDA TRANSICIÓN

Hace ya algunos años se produjo un gran cambio en los Sistemas Informáticos. Se pasó de una arquitectura basada en *mainframes*, host con terminales diseminados por la estructura de una organización, a redes locales con distintos equipos funcionando como servidores y clientes a lo largo y ancho de esa misma estructura.

La aparición de diversos Sistemas Operativos de sobremesa con capacidades de red hacen pensar en una segunda transición de sistemas Cliente/Servidor (C/S) basados en redes locales, normalmente Ethernet, a sistemas C/S totalmente distribuidos, en los que la distancia no sea un factor determinante.

LOS NUEVOS SISTEMAS OPERATIVOS

Los Sistemas Operativos (SO) estaban claramente separados en SO cliente y SO servidores. Los primeros manejaban todo lo relativo al ordenador en local, mientras que los segundos administraban los recursos compartidos y todo lo relacionado con ellos. Ahora ha nacido una serie de SO híbridos capaces de manejarse bien en ambos mundos.

LOS SISTEMAS OPERATIVOS DE RED

Los Sistemas Operativos de Red (NOS, *Net Operative System*) han tenido siempre la finalidad de ocultar la red, de hacer pensar al usuario que la red no existe. Para lograr esa transparencia de la red frente al usuario podemos hablar de ciertos matices:

- geográficos: Existe un cambio constante en la red, donde clientes y servidores se conectan y desconectan, cambiando incluso su localización. No deberíamos preocuparnos de donde se encuentra la información que esas máquinas nos proporcionan, ni como conseguirla, sólo de que existe y que podemos recuperarla, independientemente de cómo hacerlo.

- en cuanto a los espacios de nombres: Debe existir un estándar de acceso a recursos en esta red. En este sentido se desarrolla actualmente una segunda versión de *Lightweight Directory Access Protocol* (LDAP), el estándar de Internet para los servicios de directorio. (ver cuadro 1).

- en cuanto a los contenidos: debería ser posible el intercambio de recursos entre servidores sin afectar a los clientes

- en cuanto a la administración: Los NOS deben emplear un interfaz único de administración, así como integrarse con los servicios locales de administración.

- en cuanto a la conexión y seguridad: Los usuarios deben ser capaces de conectarse desde cualquier lugar de una manera segura, por ejemplo obligando a clientes y servidores a probar su identidad, recurriendo a una tercera parte de confianza para realizar esta autentificación. Un ejemplo de esto es Kerberos, desarrollado en el MIT, que es el sistema de seguridad

de su DCE *(Distributed Computing Environment)*.

- relativos a una replicación transparente: El cliente no debe conocer cómo está distribuida la información. Si un directorio está distribuido en varias máquinas, el NOS es el encargado de mantener la integridad de esa información

- en cuanto a una tolerancia a fallos: El NOS debe soportar diversos métodos de seguridad frente a fallos, frente a problemas de conexión de la red, etc.

- en cuanto a las comunicaciones: Los NOS deben ocultar complejidades, tales como el uso de múltiples protocolos, tras abstracciones para la comunicación entre procesos. Con esto me refiero, por ejemplo, al RPC (Remote Procedure Call) o similar, que permite el intercambio de datos a través de la red. (ver cuadro 2).

DETALLES DE DIVERSOS SO

Desde un punto de vista del C/S, SO como OS/2, Windows NT y UNIX cumplen unos requisitos básicos para trabajar en un entorno distribuido, como son:

a) Multitarea preventiva
b) Recursos virtuales
c) Comunicación entre procesos
d) Sincronización entre procesos, así como evitar interferencias entre procesos
e) Enlaces Dinámicos, DLL's de OS/2 y NT, y *shared libraries* de UNIX
f) Gestión de memoria compartida y virtual
g) Sistemas avanzados de archivos, como HPFS y NTFS, OS/2 y NT respectivamente
h) Gestión de excepciones (*signals* en UNIX)

Por otra parte, para realizar con facilidad el trabajo como servidor, estos SO deben ofrecer escalabilidad, y unido a esto, deben ser capaces de soportar hardware SMP (Symmetric MultiProcessing). A su vez deben ser capaces de dar servicio a múltiples clientes en distintas localizaciones de una manera transparente.

COMUNICACIÓN ENTRE PROCESOS

Al ser C/S una arquitectura distribuida, ¿cómo se realiza la comunicación entre procesos?. Ya se ha comentado que una manera común de realizar un intercambio de información es mediante RPC, pero además también existen comunicaciones punto a punto y mediante middleware basado en mensajes, MOM (Message Oriented Middleware).

- Las comunicaciones punto a punto indican que ambos participantes usan el mismo interfaz de protocolo para comunicarse. NOS de este tipo son los que poseen un *stack* nativo y emulan otros. Por ejemplo IPX/SPX es el protocolo nativo de Netware y emula otros como TCP/IP.

- Las comunicaciones a través de mensajes se basan en el envío por parte del cliente de mensajes con un formato estándar a través de la red, que se colocan en las colas de los servidores, y se van atendiendo dichas peticiones, enviando las respuestas a los clientes a través de colas de salida.

MIDDLEWARE

Este término cubre un amplio campo que cubre todo el software que hay entre clientes y servidores. En definitiva, middleware es el puente que permite la comunicación entre un cliente y su servidor.

Se puede dividir en:

- middleware general, como pilas de comunicaciones, servicios de autentificación, llamadas a procedimientos remotos (RPC), servicios distribuidos de impresión y de archivo, etc. Baste como ejemplo TCP/IP, LAN Server y LAN Manager entre otros.

- middleware específico de bases de datos (por ejemplo Open Data Base Connection, ODBC, de proceso en línea de transacciones (por ejemplo RPC), de Groupware (por ejemplo MAPI), de objetos (ORB, de OMG).

PROGRAMAS SERVIDOR Y EL CLIENTE

En un entorno distribuido existen dos grupos de programas claramente diferenciados. Por una parte están los programas servidor, que se encargan de atender las distintas peticiones que le llegan de distintos clientes en relación a un recurso compartido que éste administra. Por otra parte, están los programas cliente, de diversa naturaleza, pero que tienen en común el envío de peticiones a servidores. En función del tipo de peticiones que realiza y el interfaz, se pueden agrupar en:

1. Clientes batch: Las peticiones se generan con una participación humana mínima. Pueden necesitar o no multitarea. En el primer grupo se podrían encontrar los cajeros automáticos, mientras que en el segundo estarían los programas demonio de UNIX.

2. Clientes interactivos basados en objetos: Estos clientes generan peticiones a través de un Interfaz Gráfico de Usuario (GUI, Graphic User Interface). Esta interacción entre GUI y usuario suele ser secuencial, en la que se selecciona algún objeto y se aplican sobre él una serie de acciones predefinidas. Como ejemplo de estos clientes estarían Windows 3.1 y OSF Motif (especificación CUA 89)

Los Sistemas Operativos de Red han tenido siempre la finalidad de ocultar la red

3. Clientes interactivos orientados a objetos (OOUI, Object Oriented User Interface): Estos clientes son un 'lugar' donde se pueden integrar múltiples elementos que se ejecutan concurrentemente. Proporcionan una integración total con el entorno de trabajo, además de un acceso consistente a la información. Estos objetos están en continua comunicación unos con otros, normalmente mediante mensajes, pidiendo y ofreciendo servicios. De esta forma estos entornos necesitan un sistema de comunicaciones interactivo, en tiempo real y concurrente. Ejemplos de este tipo de entornos son OS/2, NextStep, McIntosh y Windows 95/NT.

En el (cuadro 4) se realiza un comparación de estos dos últimos, GUI y OOUI.

LOS 4 FANTÁSTICOS

Actualmente coexisten cuatro paradigmas C/S que compiten por ser la tecnología dominante en la creación de aplicaciones C/S: servidores SQL, Monitores de Proceso de Transacciones(TP, Transaction Processing), Groupware y Objetos Distribuidos.

MODELOS DE ACCESO A DATOS

Los servidores de bases de datos en entornos C/S descansan en su mayor parte en el SQL para manejar datos, con las consiguientes ventajas de un lenguaje de alto nivel, declarativo, ideal para usuarios no informáticos. Por otra parte, el modelo relacional con el que trabajan estas bases de datos separa los aspectos lógicos y físicos de la información, trabajando simplemente con la parte lógica del modelo de datos. Un servidor de base de datos permite un acceso seguro a datos compartidos, gestiona y controla la ejecución de comandos SQL, proporciona capacidades administrativas sobre los datos, mantiene catálogos dinámicos de los objetos de la base de datos, gestiona la concurrencia, seguridad, consistencia e integridad referencial de la base de datos y controla y supervisa la ejecución de transacciones sobre esa base de datos.

Como características avanzadas de los servidores de base de datos están los *trigger,* las reglas y los procedimientos almacenados.

- Un *trigger* es una acción especial definida por el usuario, que se ejecuta automáticamente en función de un evento ocurrido en la base de datos. Por ejemplo, actualizar una tabla cuando se conecte un cliente determinado al servidor de base de datos.

- Las reglas son *trigger* especiales que se emplean para realizar chequeos sobre los datos cuando sucede un evento externo, como puede ser realizar una acción sobre una tabla a una hora determinada.

- Un procedimiento almacenado es un proceso SQL que existe en el servidor y que es invocado explícitamente por el cliente o por otro proceso del servidor.

Middleware es el puente que permite la comunicación entre un cliente y su servidor

En la actualidad, trabajar con servidores de base de datos puede ser el paraíso o simplemente el mundo de las tinieblas, en función de si el proveedor de tu sistema es único o son varios.

Cuando existe una plataforma de varios proveedores puede suceder que las API SQL sean distintas, no pudiendo comunicarse entre sí diversos drivers SQL, con el uso de más recursos de lo necesario (más memoria e incluso necesidad de varios stacks), múltiples herramientas administrativas, etc. En resumen, el CAOS.

Sin embargo, en una plataforma de un solo vendedor:

- hay un API SQL propietario corriendo sobre diversas plataformas cliente

- existe un driver SQL propietario para gestionar las peticiones al API SQL propietario y el intercambio de información con el servidor

- se da soporte FAP (Format And Protocol): producto que escucha diversos stacks y los traduce al protocolo del servidor. Por ejemplo, Oracle 7 para UNIX traduce paquetes IPX/SPX (Cliente Novell) a paquetes TCP/IP (Servidor UNIX)

- existen puertas de acceso a servidores de base de datos de otros proveedores. Por ejemplo, existe un programa que se interpone entre Oracle y DB2 y que los comunica.

- existen herramientas administrativas integradas para su gestión

- existen herramientas gráficas de acceso a los datos y creación visual de OOGUI.

Para el caso de un entorno heterogéneo de base de datos pueden plantearse diversas soluciones, como puede ser un interfaz SQL común y una API común, con drivers específicos para cada base de datos. En resumen, los sistemas heterogéneos de base de datos en un sistema C/S funcionan bien, pero la dimensión de complejidad que se debe resolver para lograrlo es grande.

Un paso adelante en los modelos de datos son los almacenes de datos (Data Warehouses). La idea es proporcionar un repositorio de datos lo suficientemente amplio y rico como para permitir la ejecución de clientes inteligentes. Para extraer la información que hay en estos almacenes, que por definición son gigabytes de información, existen diversas herramientas. La más sencilla es un interfaz gráfico que te permita elegir la información que deseas a través de botones y menús, sin tener que aprender ni siquiera SQL.

Esta herramienta responde a preguntas del estilo a "¿cuál ha sido la producción de micros en el último trimestre?", y un ejemplo es Seagate, de Crystal Reports.

Esto se queda corto si lo que se necesita es una visión dinámica de los datos. Para estos usuarios aparecen las herramientas OLAP (on-line analytical processing), sobre bases de datos multidimensionales (MDDB) o, últimamente, ROLAP (relational OLAP), sobre bases de datos relacionales multidimensionales (RMDDB), que permiten un análisis de grandes volúmenes de información en periodos críticos, de 1 a 3 segundos normalmente.

Un paso más son los sistemas de soporte a la decisión, (EIS Executive Information System), que permiten a profesionales con poco tiempo disponible y sin conocimientos suficientes para ejecutar herramientas OLAP obtener la información que precisan. Proveen un acceso flexible a la información así como herramientas para visionarlos en diversos formatos (tanto en forma de texto como de gráficos) e incluso exportarlos a hojas de cálculo o procesadores de texto.

Además existen otras herramientas de análisis a las que se les provee de información y devuelven predicciones o información oculta, sin preguntas ni solicitudes explícitas. Estas herramientas usan las tecnologías informáticas más avanzadas, como redes neurales, modelos predictivos, etc. Un ejemplo de este tipo de herramientas es Intelligent Miner, de IBM.

MONITORES DE PROCESO DE TRANSACCIONES

Aparecen en mainframes para suministrar entornos de ejecución robustos, capaces de soportar una gran carga de clientes. Al migrar hacia sistemas C/S, los monitores de Proceso de Transacciones (Transaction Proccessing, TP) se tornan en parte clave de los nuevos entornos, ofreciendo interactividad, seguridad y concurrencia. Además manejan procesos y organizan programas, dividiendo la complejidad de las aplicaciones en trozos manejables, las transacciones.

Una transacción es una colección de acciones con una serie de propiedades: ACID

- Atomicity (Atomicidad): Una transacción es la unidad básica de trabajo: es indivisible. Todas sus acciones o terminan correctamente o fallan
- Consistency (Consistencia): Esto implica que después de haber ejecutado una transacción, ésta debe dejar el sistema en un estado estable, y si ésta falla, debe dejar el sistema en su estado inicial
- Isolation (Aislamiento): Una transacción no debe verse afectada por la ejecución de otras transacciones.
- Durability (Durabilidad): Esto implica que los cambios realizados con éxito por una transacción deberían permanecer incluso después de fallos en el sistema

Una transacción es una colección de acciones con una serie de propiedades: ACID

Existen distintos modelos de transacciones que definen cuándo empieza una transacción y cuándo acaba. En una transacción plana todo el trabajo se realiza en un mismo nivel, no está jerarquizada. Empieza con un begin_transaction y termina o con commit_transaction o con un abort_transaction, con la filosofía del todo o nada comentada anteriormente.

Un ejemplo de monitor de TP es X/OPEN DTP (Distributed Transaction Processing). Permite a distintas aplicaciones compartir los recursos ofrecidos por distintos gestores de recursos, coordinando su trabajo dentro de transacciones globales. Sus componentes son:

- un gestor de recursos. Es cualquier software que administra recursos compartidos, como un sistema de archivos
- un gestor de transacciones. Coordina y controla los gestores de recursos
- distintos programas de aplicación

GROUPWARE

El mercado de groupware movió en 1996 miles de millones de dólares. Groupware está cambiando la manera de comunicarse dentro de una empresa y prácticamente todos los procesos de negocio.

El groupware es una colección de tecnologías que permiten representar procesos complejos que giran alrededor de actividades cooperativas entre personas. Desde el punto de vista tecnológico, el groupware agrupa la gestión de documentos multimedia, e-mail, planificación de actividades, flujo de trabajo y conferencias.

A primera vista se podría pensar en el groupware como bases de datos SQL especializadas, pero existen claras diferencias: las bases de datos trabajan con información muy estructurada, a la que se accede mediante SQL. Las aplicaciones groupware sin embargo trabajan con información altamente desestructurada, como son textos en distintos formatos, imágenes, faxes, e-mail, BBS, etc. Además estas nuevas herramientas permiten agrupar toda esta información en nuevas entidades genéricas, denominadas documentos, así como todo lo necesario para interrogar y localizar información y poder navegar por la base de datos.

Los documentos no son más que contenedores de información heterogénea, sin una estructura fija, y son la unidad básica de gestión, como las tablas en una base de datos SQL.

Además, empleando herramientas como OpenDoc, OLE y DDE, o editores nativos, la aplicación groupware permite manipular los documentos a demanda del cliente con las herramientas con que se crearon los documentos.

Como se comentó anteriormente las herramientas groupware agrupan cinco tecnologías básicas:

1. Gestión de documentos multimedia.

2. Correo electrónico. Permite enviar y recibir electrónicamente documentos entre miembros de un grupo de trabajo de una manera sencilla, rápida y eficaz.

3. Planificación de actividades. El software groupware se encarga de sincronizar agendas, calendarios de reuniones, la situación de un proyecto de un grupo de trabajo, etc. Se trata de simplemente de organizar y planificar tareas de una manera rápida, sencilla y eficaz.

4. Flujo de trabajo. Es capaz de entender y automatizar los distintos pasos que un documento recorre en una cadena de valor añadido, enviando el documento en cada paso a la persona adecuada. Por ejemplo, si se solicita un crédito, la petición pasa por diversas fases:

- Petición de un crédito
- Análisis crediticio
- Propuesta
- Aprobación o rechazo
- Contestación al cliente
- Registro y archivo de la petición

En cada etapa la petición, convertida ya en documento, pasa por diversas manos, analizándola e incluso añadiendo nuevos elementos al documento, hasta su destino final. Los sistemas de flujo de trabajo automatizan este proceso.

Las principales características de un sistema de flujo de trabajo son:

- Cubrir las necesidades del cliente de una forma personalizada, automatizando tanto procesos deterministas como procesos menos concretos
- Integración con otras aplicaciones
- Programación con metáforas visuales
- Integración con e-mail, MOM o RPC
- Facilitar APIs que permiten a los desarrolladores personalizar la herramienta de flujo de trabajo

5. Conferencias. Se trata de reuniones no presenciales entre miembros de un grupo de trabajo, en las que la herramienta groupware se encarga de enviar la voz y/o la imagen a través de los medios telemáticos de la organización.

El Groupware está cambiando la manera de comunicarse dentro de una empresa y prácticamente todos los procesos de negocio

Como ejemplo de este tipo de herramientas groupware está Lotus Notes, actualmente principal exponente de esta tecnología. A Notes se le aplica todo lo visto anteriormente:

- como servidor de base de datos documental, almacena y coordina el acceso multiusuario a información multimedia
- como servidor de correo electrónico, permite el intercambio electrónico de información
- posee una infraestructura de seguridad servidor/servidor. Mediante un mecanismo de replicación, sincroniza copias de la misma base de datos en distintos servidores
- incorpora un entorno gráfico de usuario, GUI, capaz de mostrar distintas vistas de los documentos, un interfaz para el correo electrónico, etc.
- ofrece servicios distribuidos, incluyendo firmas electrónicas, seguridad, servicios de administración de base de datos, espacio global de nombres y gestión del sistema
- dispone de herramientas para el desarrollo de aplicaciones, como generadores de pantallas (en entorno GUI), plantillas y soporte ODBC

OBJETOS DISTRIBUIDOS

Esta es la gran apuesta para lograr una arquitectura C/S flexible y muy versátil.

En 1989 se formó un grupo de trabajo, el Object Management Group (OMG), que agrupa a más de 400 empresas del sector, que desarrolló una definición de una arquitectura que permitiera a un software abierto, de distintos vendedores, comunicarse entre sí a través de distintas redes y sistemas operativos. Este software abierto se puede ver como un bus que comunica los distintos objetos que existen en un sistema C/S y se está convirtiendo en la base para todo tipo de middleware C/S.

El OMG definió la forma de especificar una interfaz entre distintos componentes, dejando abierta la implementación de esa definición. Esta definición se escribe en IDL(Interface Definition Language), independiente de cualquier lenguaje de programación y en ella se especifican los servicios que presta cada componente: métodos, parámetros, tipo de errores que devuelven e incluso las relaciones de herencia con otros componentes.

A este bus que comunica distintos objetos se le llama ORB (Object Request Broker) y no es más que un middleware que establece una relación C/S entre objetos. Un cliente

puede invocar un método de un objeto servidor remoto de forma transparente, ya que es el ORB el encargado de buscar el servidor que ofrece el servicio solicitado, de pasarle los parámetros indicados y de devolver los resultados al cliente, simplemente usando IDL, sin necesidad de conocer el lenguaje de codificación el servidor.

CORBA

CORBA (Common Object Request Broker Architecture) es un conjunto de especificaciones aprobadas por el OMG que definen un esqueleto ORB a ORB basado en TCP/IP. Estas especificaciones también permiten a los componentes crear identificadores únicos para reconocer los distintos interfaces.
En cuanto al cliente, CORBA define:

- Protocolos IDL: proporciona una interfaz estática con los servicios de objetos, generada con el compilador IDL
- APIs de invocación dinámica: permiten al cliente acceder a un método en tiempo de ejecución. CORBA define APIs estándar para saber qué servicio se está solicitando, generar los parámetros, invocar los métodos y devolver los resultados
- APIs de interfaz con el repositorio de implementación: a través de estas API obtenemos descripciones de las clases registradas en el sistema, los métodos que soportan y los parámetros que requieren. CORBA llama a estos métodos firmas (method signatures). Estas API nos permiten acceder al repositorio, que es una base de datos que contiene las descripciones IDL compiladas
- Interfaz ORB: APIs que permiten gestionar servicios locales a la aplicación, pero que no forman parte del mecanismo de llamadas C/S

Y en cuanto al servidor, CORBA define:

- Protocolos IDL, que proporcionan las interfaces estáticas de cada servicio exportado por el servidor. Éste no distingue entre llamadas estáticas o dinámicas, son las herramientas IDL las que crean y mantienen esas llamadas
- El adaptador de objetos, que interacciona con los servicios básicos de comunicaciones del ORB y acepta peticiones de servicio para los objetos del servidor. Nos ofrece un entorno de ejecución para instanciar objetos del servidor, pasarles peticiones y asignarles identificadores (referencias a objetos en CORBA). También registra las clases que soporta y sus instancias en ejecución, dentro del repositorio de implementación
- El repositorio de ejecución: nos da información en tiempo de ejecución de qué clases soporta un servidor, qué objetos están instanciados y sus identificadores. También se emplea como lugar común donde guardar la información relacionada con la implementación de un ORB (información de auditoría, seguridad, estadísticas, etc.)
- Interfaz ORB: al igual que en el caso del cliente, son APIs para la gestión de servicios locales

A pesar de esto, un ORB por sí solo no tiene todo lo necesario para que los objetos interoperen a nivel de aplicación. El ORB es como una línea telefónica, proporciona la conectividad básica. El resto de servicios los dan objetos por encima del ORB con interfaces IDL. Ejemplo de estos servicios pueden ser la persistencia, que permite a un objeto existir más allá del proceso que la creó (guardarlo en disco), los eventos (envío de notificaciones de sucesos a objetos registrados como interesados), la convención de nombres (paso de nombres legibles por una persona a identificadores generados por la máquina), transacciones (extiende las propiedades ACID a los objetos), o propiedades (se nos dan atributos que se pueden nombrar dinámicamente asociados a un objeto).

A este bus que comunica distintos objetos se le llama ORB

DOCUMENTOS COMPUESTOS

En la línea de CORBA aparecen herramientas que aprovechan esta tecnología y nacen los documentos compuestos, ya mencionados anteriormente, como contenedores de información de diversas fuentes, de modo que el usuario puede interactuar con cada uno de esos componentes.

Los dos estándares que compiten hoy en día en este apartado son OLE, de Microsoft, y OpenDoc, de diversas compañías, entre ellas IBM y Apple.

Por una parte, OLE se basa en COM (Common Object Model), la alternativa Microsoft a CORBA, mientras que OpenDoc se basa en la implementación CORBA de IBM llamada System Object Model (SOM).

En principio OLE y OpenDoc son incompatibles, palabra maldita de la informática, aunque la tendencia es hacia el entendimiento, como en la última versión de OpenDoc por parte de IBM, capaz de incrustar componentes OLE.

HERRAMIENTAS DE DESARROLLO

Existen un gran número de herramientas de desarrollo para entorno C/S, como son generadores visuales de aplicaciones (Delphi), lenguajes estándar de alto nivel (VisualAge for C++), lenguajes propietarios de alto nivel (Powerbuilder) o acceso a bases de datos relacionales propietarias (Visual Basic y Access).

La última herramienta RAD (Rapid Application Development) para entornos C/S de Borland ha sido C++ Builder 1.0. Ésta es muy similar a Delphi, pero con generación de código C++.

CONCLUSIONES

Como en otros campos cabe la duda de cual será el paradigma vencedor en esta lucha por la supremacía en entor-

Cuadro 1: LDAP

LDAP pretende ser la respuesta al acceso a información y aplicaciones en Internet. Es una mejora sobre X.500, el protocolo OSI para la administración de recursos y directorios. Proporciona a los desarrolladores una plataforma común sobre la que trabajar con recursos, independientemente del vendedor. Se basa en el DAP (Directory Access Protocol), desarrollado para la X.500, y que mejora la manera de acceder y actualizar la información de los directorios en un modelo C/S.

Cuadro 2: RPC

RPC nació en SUN como parte de NFS y de su Open Network Computing (ONC). Más tarde también la Open Software Foundation (OSF) lo incorporó a su DCE. RPC permite que aplicaciones en distintas máquinas intercambien información de una manera sencilla. RPC te permite llamar prácticamente a cualquier método que exista en la red. Se pueden pasar argumentos, y recibir los resultados de la función, ejecutada incluso en el servidor. Curiosidad: El interfaz de programación de RPC está implementado como un generador de código C.

Cuadro 3: Funcionamiento básico de ODBC

El programa realiza las peticiones a distintas fuentes de datos a través del API de ODBC. Este traduce la petición en función del tipo de fuente de datos usando el driver adecuado de ODBC, y la envía al servidor indicado, que realiza la operación pedida y, si es el caso, devuelve la respuesta al programa a través de ODBC.

nos distribuidos C/S, pero la tendencia general a medio y largo plazo son los objetos distribuidos, la tecnología más puntera, y a más corto plazo el Groupware, con Lotus Notes a la cabeza.

Dentro de los objetos distribuidos la lucha entre CORBA y Network OLE (Microsoft) será grande y el mercado decidirá, como siempre, el ganador de esta penúltima batalla.

BIBLIOGRAFÍA

- Baker, R.H., Networking the Enterprise. How to Build Client/Server Systems that Work
- Berson, A., Client/Server Architecture
- Bitterer y otros, VisualAge and Transaction Processing in a Client/Server Environment
- Boar, B.H., Cost-Effective Strategies for Client/Server
- Bochenski, B., Implementing Production Quality Client/Server Systems
- Vaughn, L.T., Client/Server System Design and Implementation

DIRECCIONES DE INTERÉS

- Object Management Group OMG http://www.omg.org
- Open Software Foundation OSF http://www.osf.org
- IBM http://www.ibm.com
- LDAP http://www.ietf.cnri.restom.va.us/html.charters/asid-charter.html
- Borland http://www.borland.com

CUADRO 4.

Estructura de la aplicación	Ventanas principal y secundarias, una barra de menú e iconos	Colección de objetos que cooperan entre sí. Cada objeto tiene una representación gráfica, un icono
Iconos	Representan una aplicación en ejecución	Representan los objetos del sistema
Iniciar Aplicaciones	Inicias una aplicación antes de seleccionar un objeto para trabajar con ella	Al abrir un objeto se inician las aplicaciones asociadas con él
Ventanas	Es la manera gráfica de interaccionar con los objetos	La ventana es la puerta que nos deja ver lo que hay dentro de un objeto, en una relación de 1 a 1
Aplicaciones Activas	Se muestran en su modo de icono (minimizadas)	Se destaca el icono de la aplicación en uso
Manipulación Directa	Especifico de cada aplicación	Se basa en arrastrar y soltar objetos. Es proprocionado por el sistema a todos los objetos
Acciones	Primero se elige el objeto y luego la acción a través del menú	Se usan las facilidades de arrastrar y soltar para realizar acciones, como arrancar una aplicación dándole como parámetro un objeto
Creación de objetos Nuevos	Especifico de cada aplicación	Se basa en plantillas, que situamos arrastrando y soltando donde queramos
Comunicación entre tareas	Se realiza a través de otras tareas que soportan OpenDoc, OLE y DDE	Los objetos se comunican entre si usando ORB, OpenDoc, OLE y DDE
Cerrar Aplicaciones	No se guarda la configuración	Se guarda en disco la configuración visual de la aplicación cuando la cerramos

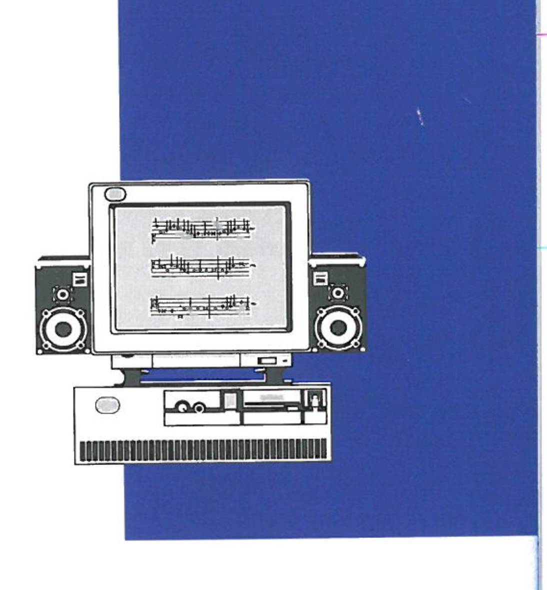
SHOCKWAVE O LA INTERACTIVIDAD EN LA WEB

Javier Sarsa Garrido

Desde los comienzos del HTML, todos hemos buscado con mayor o menor acierto aumentar la capacidad de nuestras páginas Web para atraer más usuarios o para que éstos se queden enganchados en ellas durante un tiempo más largo. La solución a esta cuestión no se presenta fácil; sin embargo, parece que ésta pasa por conseguir la participación del usuario en una mayor medida, o lo que es lo mismo, "La búsqueda de la Interactividad". Si analizamos lo que se ha venido realizando hasta ahora comúnmente, encontramos que el hipertexto y las imágenes (estáticas, animadas o mapas gráficos) ocupan un gran porcentaje de las páginas existentes. Las compañías de software saben que será la interactividad la que conseguirá el despegue definitivo de Internet para la globalidad de los usuarios. Así han aparecido diversas técnicas con las que incorporar esta capacidad de comunicación: Javascript y más aún Java ya han sido mayoritariamente aceptados como una buena estrategia de futuro, pero no hay que olvidar que otras compañías han puesto a nuestra disposición nuevas formas de enriquecer nuestros nodos; estoy hablando de la tecnología SHOCKWAVE.

¿QUÉ ES SHOCKWAVE?

Para aquéllos que no habéis oído todavía este término os puedo decir que, desde sus comienzos, Shockwave ya permitía incorporar la multimedia real en Internet. Shockwave es una familia de módulos (*plug-ins*) que permiten ejecutar en las páginas Web los documentos multimedia generados por los programas de la casa Macromedia. La importancia de esta posibilidad radica en que, a diferencia de los productos similares de otras casas, las aplicaciones de Macromedia están ampliamente consolidadas entre los desarrolladores de multimedia. Desde que fue presentado en 1995, más de 17 millones de módulos han sido descargados del nodo de Macromedia y empresas como General Motors, Nissan, Kodak, Microsoft, Intel o Apple, incluyen contenidos basados en esta tecnología. Así, podemos desde entonces disponer de módulos que cubran nuestras necesidades de multimedia avanzada (Director y Authorware), de sonido (Soundedit), de gráficos vectoriales (Freehand) o de gráficos bitmap de alta resolución (xRes).



¿DÓNDE FUNCIONA SHOCKWAVE?

Por el momento existe la posibilidad de emplearlo en las plataformas Windows y Macintosh, con cualquiera de sus S.O.: Windows 3.1, 3.11, 95, NT 3.5.1 y NT 4.0 en el caso del PC y MacOS 7.5.3 (recomendado) o superior en el del Macintosh.

Además lo soportan casi todos los navegadores más usuales: Netscape Navigator 2.02 ó 3.0 final, MS Internet Explorer 3.0 final, Attachmate's Emissary, Netmanage's Websurfer y America Online 3.0 (sólo Director y audio), todos ellos disponibles en ambas plataformas.

¿QUÉ NECESITAMOS TENER INSTALADO?

Puesto que se trata de módulos, la primera de las operaciones para poder ver las páginas *"shockwaveadas"* consistirá en descargarlos de la dirección 'www.macromedia.com'. Una vez instalados en nuestra carpeta de *"plug-ins"*,

La tecnología Shockwave complementa otras formas de interactividad como las que proporcionan Java o JavaScript. Desde sus comienzos ha permitido incorporar la multimedia real en Internet

deberíamos asegurarnos de que nuestro navegador o nuestro servidor está configurado para gestionar los tipos de fichero correspondientes. (Ver tabla 1)

SHOCKWAVE PARA DIRECTOR

Director es, en el mercado de producción multimedia, la herramienta más

TIPOS MIME	EXTENSIONES
application/x-director	dir, fgd, dxr, dcr
application/x-authorware-map	aam
application/x-authorware-seg	aas
application/x-authorware-bin	aab
application/x-freehand	fh4, fh5, fhc, fh

TABLA 1: Configuración de los tipos de fichero en el navegador.

utilizada, por ello nos centraremos en la parte de Shockwave que gestiona sus documentos. Este apartado supone que el lector está familiarizado en parte con la aplicación Director y con su lenguaje de programación llamado Lingo.

		VELOCIDAD DE TRANSFERENCIA DISPONIBLE			
TIPO	TAMAÑO	14.4 kbps	28.8 kbps	64 kbps	1.5 mbps
Animación pequeña	30 K	30 seg.	10 seg.	6 seg.	1 seg.
Peq. película completa	100-200 K	100/200 seg.	50-100 seg.	20-40 seg.	1 seg.
Pequeño video-clip	500 K	500 seg.	120-240 seg.	90 seg.	3 seg.
Película completa	1 Mb	No acep.	No acep.	180 seg.	6 seg.
Vídeo continuo MPEG	Indefinida	No acep.	No acep.	No acep.	Continuo

TABLA 2: Tiempo de descarga aproximado de algunos documentos.

Cuando nos enfrentamos, como programadores, a la tarea de tener que introducir un documento de Director (o película) en nuestra página Web, debemos tener en cuenta que el flujo de información que atraviesa la red siempre que un usuario solicita un documento de multimedia complejo, puede ser muy grande. (Ver tabla 2)

Para evitar en gran medida este problema se recurre como era de esperar a la compresión. Las películas de Director (.dir) pueden ser tratadas con un compresor especialmente diseñado para ellas denominado *Afterburner para Director*. Se incluye como una opción bajo el menú Xtras de Director 5.0 y está disponible como aplicación separada para Director 4.0. El resultado de esta operación será una película compactada (.dcr) que se comporte en la red como la original, pero cuyo tiempo de carga será mucho menor.

Y vamos con información sólo para programadores. En este caso existen dos lenguajes diferentes que deben combinarse. Por un lado han sido incorporados en Director diversos comandos de Lingo que permitan ordenar acciones al navegador desde nuestra película. En la otra parte, podemos pasar parámetros desde las páginas de Web a nuestra película.

Iremos, en primer lugar, a esta última opción. Como otros elementos, las películas son "embutidas" en las páginas Web con el tag de **HTML** "embed".

```
<EMBED src="path/fichero.ext" [width=x]
[height=y] [palette=ground] [bgcolor=color]
[pluginspace ="url"] [param1=valor1]
[param2=valor2] .... [paramN=valorN] >
```

src especifica el camino a nuestro documento (.dir, .dxr o .dcr)
width y height se utilizan para dar a conocer las dimensiones de la película en la página.
palette permite controlar la paleta de colores que utilizará el navegador.
Palette=foreground empleará la paleta de la película de Director, pudiendo afectar a otras imágenes de la misma página. (Foreground no es soportado por Internet Explorer 3.0).
Palette=background, que es el parámetro por defecto, utilizará la paleta del sistema.
bgcolor para decidir el color de fondo del documento embutido.
pluginspace abriría un url para el usuario que no tuviera instalado Shockwave.
"http://www.macromedia.com/shockwave"

param1,param2,... son parámetros adicionales sin nombre definido, que podríamos añadir para comunicarnos con nuestra película.
El segundo lenguaje a considerar, la utilización de comandos desde Director para comunicarse con el navegador, existe gracias a la incorporación de varias instrucciones en **Lingo**. Éstas pueden dividirse en:

Recuperación de los parámetros pasados desde html

Suponiendo que en nuestra página HTML hubiésemos escrito:

```
<EMBED src="misdoc/peli1.dcr" width=400
height=300 volumen=5 casillasX=8 casillasY=4>
```

externalParamCount () devuelve el número de parámetros intercambiados en un entero.
Ejemplo: put externalParamCount() into numparam
Resultado: numparam contendrá 6.

externalParamName(n) devuelve el nombre identificador del parámetro número n.
Ejemplo: put externalParamName(4) into nombreparam
Resultado: nombreparam contendrá "volumen".

externalParamValue(n) nos da el valor del parámetro número n.
Ejemplo: put externalParamValue(4) into valparam
Resultado: valparam contendrá 5.

OPERACIONES ASÍNCRONAS

getNetText url La película de Director hace que el texto especificado por url se cargue en memoria.
Ejemplo: getNetText "http://servidor.es /pagina.txt"
(Ver también NetDone y NetTextResult)

preloadNetThing url carga el archivo referenciado por url en la caché para su uso
posterior. La película actual sigue ejecutándose.

gotoNetMovie url Carga una nueva película de Director definida en url y la ejecuta cuando termina de cargarse en la misma área, cerrando la anterior que estaba en curso.

gotoNetPage url, target
Carga un nuevo documento especificado en url y lo coloca en la ventana especificada en target. Como en HTML, target puede ser "_blank", "_top", "_self", "_parent", "nombre de frame" o "nombre de ventana", dependiendo la ubicación final que deseemos para el documento.
Ej.: gotoNetPage "http://servidor.es /pagina.html", "_blank" abrirá el contenido solicitado en una nueva página.

netDone () Devuelve "false" mientras una de las operaciones anteriores está en proceso y "true" cuando esta ha finalizado.
Ejemplo: if NetDone()=true then beep

netError () Cuando la operación de transferencia ha finalizado devuelve
"OK" o un error si ésta ha fracasado. También devuelve "empty" durante la transferencia y "None" si no ha existido ninguna operación.

netAbort. Cancela la operación de red en curso.

OPERACIONES DE RECUPERACIÓN

netTextResult () Devuelve el texto capturado por getNetText.

netMIME ().Devuelve el tipo MIME del documento HTML cargado.

netLastModDate (). Devuelve la fecha de última modificación recogida de la cabecera "date last modified" del documento cargado.

getLatestNetID (). Devuelve un identificado único para la última operación asíncrona solicitada.

netStatus msg Altera el contenido de la barra de status del navegador con el mensaje msg.
Es equivalente al javascript window.status=msg.
Ej.: netStatus "Esté atento a lo que va a aparecer."

ALMACENAMIENTO DE LAS PREFERENCIAS DE USUARIO

setPref prefName, prefValue. Escribe exclusivamente en la carpeta de plug-ins del disco duro del usuario, un fichero de nombre especificado en prefName.
PrefValue es el string que queremos escribir.
Este procedimiento equivaldría al establecimiento de cookies.

GetPref (prefName) Devuelve el contenido del fichero especificado en prefName.

UN CASO PRÁCTICO

Imaginemos que hemos creado una película de Director consistente en el juego del puzzle clásico de mover piezas en una matriz de n x n casillas. Los datos relativos a la programación Lingo necesaria para esto no van a mostrarse. Sin embargo, sí vamos a ver las instrucciones que habría que colocar en cada uno de los dos listados para establecer la comunicación entre ambos documentos. Siguiendo en el ejemplo propuesto, podremos distribuir esta película del puzzle de manera que cualquiera que deseara embutirla en sus páginas, pudiese cambiar el lugar de la casilla en blanco (variable posvacio), el tiempo permitido para resolver el puzzle (tiempo) o el nombre de las páginas a las que el usuario accederá como premio (paginabien) o como castigo (paginamal) al éxito o fracaso en la resolución del puzzle.

Así que tenemos cuatro parámetros que escribir en nuestro HTML, aunque cabe la posibilidad de que sean opcionales si lo hemos contemplado en la película.

```
<EMBED SRC="puzzle.dir" WIDTH="308"
HEIGHT="301" ALIGN="BOTTOM" tiempo="100"
posvacio="8" paginabien="exito.htm"
paginamal="fracaso.htm">
```

En el otro listado, en el de Director, escribiríamos lo siguiente para recoger estos cuatro datos, previniendo unos valores por defecto para el caso de no contar con dichos parámetros.

```
ON STARTMOVIE
global vacio,tiempovar,bienhtm,malhtm

 — Recogida de los parametros transmitidos desde
HTML
  repeat with i=1 to externalParamCount()
  put externalParamName(i) into nombreparam
  if nombreparam="posvacio" then
put integer(externalparamvalue(nombreparam))
into vacio
  else
if nombreparam="tiempo" then
put integer(externalparamvalue(nombreparam))
into tiempovar
  else
if nombreparam="paginabien"
then  put externalparamvalue(nombreparam)
into bienhtm
  else
if nombreparam="paginamal"
then  put externalparamvalue(nombreparam)
into malhtm
  end repeat

  — Valores por defecto si no son suministrados
desde la pagina de HTML

  if voidp(vacio) then put 8 into vacio
  if voidp(tiempovar) then put 120 into tiempovar

END STARTMOVIE
```

Otras instrucciones que se han colocado en la película son las que envían al navegador a las páginas de éxito y fracaso con gotoNetPage.

```
ON SEGUNDOMENOS
 — Comprueba si el tiempo está agotado y va a la
página de fracaso si existe.
  global tiempovar,malhtm
  ...
  if tiempovar=0 then
```

```
    — Va a la pagina de fracaso si la hemos pasado como parámetro
    if voidp(malhtm) then alert "Lo siento el tiempo ha finalizado"
    else gotoNetPage malhtm
    ...
  end if
  ...
END SEGUNDOMENOS

ON COMPROBARFINAL
  — Comprueba si la disposición de las fichas es la correcta
  — y va a la página de éxito si la hemos definido.

global listafichas, listaordenada, bienhtm, resuelto
  if listafichas=listaordenada then
    ...
    — Va a la pagina de exito si la hemos pasado como parámetro
    if voidp(bienhtm) then
      alert "Enhorabuena, la resolución es correcta."
      ...
    else gotoNetPage bienhtm
  end if
END COMPROBARFINAL
```

También se ha utilizado la instrucción netStatus para dar al usuario mensajes de ayuda durante el juego.

```
ON IDLE
  — Se ejecuta cada sesentaavo de segundo si es posible, para tareas repetitivas.
  — Mira sobre que ficha está el ratón para dar un mensaje en la barra de status.
  global tiempovar
  if tiempovar=0 then
    netStatus "El tiempo ha finalizado."
    exit
  end if
  repeat with i=1 to 9
    if rollover(i) then
        if the name of cast (the castNum of sprite i)="vac"
then
        netStatus "Esta casilla no es una ficha."
      else
netStatus "Pulsa para mover la ficha."
      exit
    end if
  end repeat
  netStatus "Preparado para actuar."
END IDLE
```

SHOCKWAVE PARA FREEHAND

Analizaremos ahora con mayor brevedad y como último ejemplo, el módulo Shockwave para Freehand 7.0. En esta ocasión, la aplicación de Macromedia es un programa de diseño, basado fundamentalmente en el tratamiento de gráficos vectoriales. No existe, por tanto, lenguaje de programación.



Ejemplo práctico incluido en el CD-ROM.

Ahora, existe bajo la opción "Xtras", del menú "Window", el concepto "URL editor", que nos abrirá una ventana con la lista de URLs disponibles. Desde esta ventana podremos añadir, duplicar, eliminar, modificar o buscar URLs y relacionarlos por arrastre con elementos no agrupados del dibujo. También, bajo el menú "Xtras", se esconde un submenú llamado "Afterbuner", que al igual que antes, nos permitirá la compresión y descompresión de los dibujos para su uso en la Web.

Por último, la instrucción a añadir a la página en la que incorporar los gráficos será:

```
<EMBED src="path/fichero.ext" [width=x] [height=y] [toolbar="bottom"]>
```

donde el único parámetro nuevo es toolbar que puede tomar los valores "top" y "bottom" indicando donde aparecerá la botonera del gráfico. En esta botonera el usuario tendrá la posibilidad de moverse en el dibujo, hacer un zoom, etc.

CONCLUSIÓN

Frente a otras alternativas como Java y Javascript, la tecnología Shockwave puede constituir una buena opción en la búsqueda de la interactividad de la que hemos estado hablando; más aún cuando los lenguajes como Java no han sido desarrollados por completo. Además contaremos con la ayuda de herramientas de autor, específicamente diseñadas para gestionar elementos de multimedia, lo que hará más fácil nuestra tarea. Hablaremos más a fondo sobre este tema en futuros artículos.



UNIVERSAL SERVER DE INFORMIX

Carlos Urbano

Las empresas necesitan sistemas de información con los que llevar el control de todos sus recursos, los sistemas antiguos tenían una serie de desventajas como eran la redundancia e inconsistencia de los datos, problemas de concurrencia, problemas de seguridad, problemas de integridad y coherencia de los datos y dependencia con respecto a los tratamientos. En una base de datos actual, el conjunto de datos almacenados en memoria secundaria son accedidos directamente y manipulados por un conjunto de programas.

La base de datos como tal debe ser completa, abarcar la totalidad de los datos que la empresa necesita y soportar una estructura de los datos que permita agruparlos según una relación natural entre ellos. Además debe tener el menor número de errores posible y estar estructurada según un modelo de datos. Un modelo de datos es un conjunto de herramientas que se utilizan para describir los datos que conformarán todas los posibles recursos, así como sus relaciones, su significado y las restricciones del grupo de datos.

El entorno en el que la base de datos tiene su espacio de desarrollo y manipulación es el sistema de gestión de la base de datos. Sus componentes son variados y abarcan desde el tipo de datos y el hardware, hasta los usuarios que varían en su interrelación con el sistema de gestión.

Los SGBDS trabajan a varios niveles y están formados por un conjunto de programas que suministran a los usuarios del sistema una forma de manipular y gestionar los datos incorporados en la Base de datos, intentando asegurar su confidencialidad y seguridad.

MODELOS DE DATOS

Para representar la información de forma independiente de los sistemas de bases de datos, asegurar su transportabilidad a otros sistemas y conseguir que los datos sean fácilmente comprensibles por aquellas personas encargadas de manipular la información, se usan distintas técnicas. En la actualidad el modelo relacional es el que se utiliza de manera más frecuente para conseguir abstraer la realidad y configurar un SGBD.

El modelo relacional tiene un fundamento matemático y un álgebra (conjunto de operaciones definidas sobre los elementos del modelo) en el que no entraremos por no ser objetivo del presente artículo, adecuado para el tratamiento de los datos simples (enteros, numéricos de punto flotante, cadenas de caracteres, fechas, etc.) que proporciona métodos de consulta y acceso y permite la realización de transacciones (utilizando por ejemplo el lenguaje SQL) a alta velocidad pero no contempla la gestión de datos complejos.

Las constantes mejoras en el hardware, el incremento de la potencia de la CPU, la capacidad de la memoria y del disco, la mayor rapidez en las conexiones a redes interconectadas con mayores anchos de banda, han tenido como resultado la necesidad de manejar tipos

Las estructuras de datos aumentan su complejidad a medida que las empresas manejan información más variada y rica en contenidos. El acceso y la gestión de estos datos, imágenes 3D, audio, vídeo, datos espaciales, series temporales, etc., y de la información que está asociada a los mismos requiere tecnologías innovadoras

de datos complejos para gestionar la nueva clase de información que se obtiene gracias al avance del rendimiento de los ordenadores. El mundo empresarial genera estos nuevos tipos de datos, los medios audiovisuales y escritos, editoriales, radio, televisión, producción cinematográfica y de desarrollo software para el entretenimiento, etc. están descubriendo en la tecnología digital un medio de potenciar sus productos y competir en el mercado.

La gestión de los datos manipulados en esta tecnología no puede ser realizada mediante los SGBDR tradicionales ya que el diseño en que se basan no los trata. Lo mismo sucede con la Web, se necesitan formas de gestionar datos complejos como los que utilizan y generan los servidores Web. Las aplicaciones en tiempo real demandan una mayor información, series de tiempo, medias aritméticas, imágenes, etc.

BASES DE DATOS ORIENTADAS A OBJETOS

Como resultado de la necesidad de manejar nuevos tipos de datos y crear las operaciones necesarias para manipularlos, se aplicó la metodología orientada a objetos en la concepción de las bases de datos.

En los SGBDR se utilizó dicha metodología de abstracción de la realidad soportando de manera limitada datos complejos, almacenándolos como BLOBs (*Bynary Large Objects*), objetos binarios de gran tamaño, los cuales no pueden ser buscados, manipulados o indexados en el servidor.

La inclusión de BLOBs en las bases de datos relacionales es útil cuando se trabaja sobre disco directamente en la recuperación de archivos concretos, por ejemplo en sistemas de almacenamiento de vídeos o imágenes, pero cuando necesitamos una aplicación completa, la utilización de BLOBs limita la aplicación a la realización de funciones simples, como el borrado o la visualización de los datos.

Los BLOBs se mantienen al margen del sistema de gestión de datos en la mayoría de las funciones que éste realiza, como en la optimización de consultas o los métodos de acceso.

Para franquear esta limitación, se fue más allá en los SGBDOO. El concepto de objeto se aplicó en estos sistemas de bases de datos: los objetos tienen una identidad, el sistema genera un identificador para el objeto, que debería mantener, y además poseen un estado y un comportamiento asociados.

Los SGBDOO suelen utilizar el modelo de datos del lenguaje de programación que sirve como lenguaje de acceso a los datos y como lenguaje de desarrollo de los programas del sistema; con frecuencia se utilizan el C++ y el SmallTalk. Todos los sistemas de bases de datos actuales permiten que muchos usuarios trabajen con el sistema al mismo tiempo y para ello utilizan una arquitectura cliente-servidor.

Los datos se tratan como extensiones, donde se integran texto, imágenes, vídeos, páginas HTML y datos de las operaciones que con ellos se pueden realizar

En los SGBDOO las arquitecturas cliente-servidor suelen implementarse de una de las siguientes maneras (que proponen David DeWitt y David Maier):

- Arquitectura objeto-servidor. El objeto sirve como unidad de comunicación entre el cliente y el servidor y es éste el que concentra la mayor parte de la funcionalidad del sistema de base de datos. Como consecuencia el diseño del servidor es complicado.
- Arquitectura página-servidor. La unidad de comunicación es la página. La complejidad del diseño la tiene el cliente y así el servidor se optimiza pudiendo soportar varios clientes.
- Arquitectura fichero-servidor. Se utiliza un servicio de ficheros remotos para leer y escribir en las páginas de la base de datos. El cliente soporta la mayoría de las funciones.
- Hoy por hoy existen entornos de desarrollo estables que aplican la orientación a objetos para la creación de aplicaciones. Sin embargo, los SGBDOO son adecuados para tratar tipos de datos complejos que se estructuran según la metodología POO y son manipulados mediante lenguajes de programación orientados a objetos. Las aplicaciones que se desarrollan para estos sistemas se basan en la utilización de punteros a través de los cuales se accede a los datos y su mantenimiento resulta muy caro.

Los SGBDOO disponibles en el mercado no escalan fácilmente y se utilizan generalmente sobre pequeños conjuntos de usuarios que realizan manipulaciones complicadas de objetos, como en el caso de la manipulación de dibujos de CAD, pero no son recomendables para realizar la gestión bases de datos de un conjunto de usuarios creciente que maneja un volumen de datos también en crecimiento debido a la dificultad que entraña la forma de acceso a los mismos.

BASES DE DATOS RELACIONAL-OBJETO

Como resultado de las carencias que las bases de datos orientadas a objetos presentan en cuanto a la gestión de

contenidos no estructurados, desde Informix Software Inc. se nos propone una solución que se basa en la gestión de datos complejos de forma inteligente, permitiendo al servidor la toma de decisiones para optimizar el acceso a dichos datos, no en forma de BLOBs, sino como objetos nativos del sistema.

Las definiciones de funciones en el SGBDRO incluyen instrucciones sobre el coste de E/S y de CPU de la función, lo cual permite escoger un plan de consulta más adecuado. Las funciones se ejecutan en el servidor, cerca de los datos, lo que aumenta su rendimiento porque no hay que sacar los datos del servidor para manipularlos.

El desarrollo de aplicaciones que se basan en la información que contienen se hace posible cuando el sistema de gestión de base de datos integra las estructuras de los datos (objetos), así como las operaciones que con ellos se pueden realizar, la forma de indexarlos, la forma de optimizar las consultas sobre ellos, etc.

Con Universal Server los clientes pueden definir sus aplicaciones estableciendo la estructura de los datos que necesiten y su funcionamiento. Se proporciona al sistema de gestión la capacidad de gestionar los nuevos tipos de datos de forma que sea el contenido de la información el que defina que todas sus partes estén interrelacionadas.

Para que nos aclaremos, los datos se tratan como *extensiones,* donde se integran texto, imágenes, vídeos, páginas HTML, datos de las operaciones que con ellos se pueden realizar, entre otras informaciones que puedan estar relacionadas.

Por ejemplo, una compañía de distribución de pizzas podría gestionar su información de forma que todos sus componentes estuvieran vinculados, los pedidos, los mapas de carreteras y de calles donde se extendiera su influencia, los datos personales de sus trabajadores, los menús, las imágenes digitalizadas de sus menús y de los partes de accidentes, cuando los hubiera, los estudios de mercado, etc.

Los BLOBs se mantienen al margen del sistema de gestión de datos en la mayoría de las funciones que éste realiza, como en la optimización de consultas o los métodos de acceso

Para definir estas *extensiones*, Informix, de la mano de Illustra (origen de la tecnología en que se basan los DataBlades), propone una nueva forma de objeto denominado DataBlade TM, y que se caracteriza por tener una entidad, con atributos, métodos, estados y transiciones entre estados.

Esta nueva concepción de un objeto es utilizada en Universal Server para proporcionar a los SGBDR una manera de trabajar bajo el paradigma de la POO. De esta forma, el sistema de gestión de la base de datos tendrá el tipo de datos perfectamente definido (sea un documento, un vídeo, una imagen o un archivo de sonido).

Desde el punto de vista de un programador, un módulo DataBlade es un componente Software reutilizable que incorpora el concepto de objeto al sistema de gestión de bases de datos relacional, extendiendo la funcionalidad del servidor para que le sea posible manejar tipos de datos nuevos y complejos.

Los DataBlade se integran en el gestor de base de datos y posibilitan el conocimiento de un dominio concreto y de las funciones necesarias para que cada tipo de datos sea soportado en el propio gestor: la estructura del objeto, los métodos de acceso para las operaciones de recuperación, almacenamiento y visualización y otras funciones asociadas.

La gestión de la base de datos se basará en un interfaz SQL destinado al desarrollo y mantenimiento de aplicaciones. Las nuevas extensiones de datos podrán ser incluidas como objetos de consultas SQL, y las consultas podrán ser sensibles al contenido, es decir, hacer búsquedas o selecciones según la estructura de los objetos que definen la información. Por ejemplo, decidir cuáles de los respartidores de pizza son apropiados para llevar un encargo a una empresa según si habitualmente llevan barba.

DATABLADES: OBJETOS Y FUNCIONES

"Los DataBlades definen la estructura de un objeto de información, además de las funciones asociadas a ese objeto y los métodos de acceso para su recuperación y visualización." Esta es la definición que Informix da de DataBlade, y también puntualiza sus ventajas:

La gestión mediante DataBlade se realiza a través de módulos que desarrollan diferentes empresas además de la propia Informix

- El sistema de gestión podrá optimizar cualquier consulta que haga referencia al objeto de información, permitiendo que el rendimiento global del sistema no se vea afectado sea cual sea la complejidad de la estructura de los datos con los que trabajemos.

- A la hora de desarrollar aplicaciones basadas en los objetos, no es necesario utilizar nuevos estándares ni lenguajes de programación especiales.

- Se pueden integrar perfectamente unos con otros permitiendo así desarrollar aplicaciones multimedia basadas en el contenido de la información, dinámicas e interactivas. En un único Universal Server se pueden cargar simultáneamente muchos módulos DataBlade y los usuarios pueden extender e incluso sustituir las funciones de estos módulos, según sus necesidades.

METODOLOGÍA POO EN LOS DATABLADES

La concepción de DataBlade implica que Universal Server tenga las siguientes características:

- Universal Server permite la definición de nuevos tipos de datos, totalmente nuevos o basados en otros ya existentes; un tipo puede incluir otros tipos y hacer referencia a los mismos como parte de su definición. Así, los usuarios pueden crear tipos diferentes, conjuntos, multiconjuntos, listas, matrices, identificadores de objetos (referencias a otros objetos), que además podrán formar parte de otros tipos, y otros, según sus necesidades y completamente adaptados a la información que requieran gestionar. Esto se traduce en la posibilidad de creación de nuestras propias estructuras de datos, basadas en otros objetos ya definidos de forma que hereden sus propiedades y funciones, o bien empezar de cero, adaptándose a nuevos datos.

Universal Server permite la definición de nuevos tipos de datos, totalmente nuevos o basados en otros ya existentes

- Los tipos basados en otros tipos pueden copiar su definición o heredarla, se soporta la herencia, permitiendo que un tipo recoja partes de su estructura y funcionamiento de otro tipo predefinido.

- Pueden heredarse la estructura, las reglas (*triggers* en Universal Sever), los operadores, las funciones y las expresiones de agrupación, manteniendo así las ventajas de la OO.

Informix ha diseñado un programa para desarrolladores de módulos que facilita la construcción de nuevos módulos de software

- Los tipos están definidos completamente, es decir tipos diferentes no pueden compararse directamente, debiéndose realizar un *cast* para poder realizar la comparación, se soporta así la definición de funciones de conversión.

- Las funciones definidas por el usuario en Universal Server se corresponden con los llamados métodos en OO. Las funciones pueden ser implementadas en lenguaje C, en Java o en SQL y están limitadas por las capacidades del lenguaje utilizado, de esta forma, si se utiliza Java, se podrá realizar cualquier tarea que Java permita. Un usuario puede definir cualquier función para ser ejecutada en el servidor o en el cliente. En la definición de la función se instruirá al sistema sobre los costes de E/S y de CPU para que las consultas donde dicha función aparezca puedan ser optimizadas.

- Se admite la sobrecarga, lo que en OO equivale al polimorfismo. Las funciones se componen de un nombre y una lista de parámetros, los cuales, junto con sus tipos, forman el identificativo de la función (signature). La sobrecarga de funciones permite que pueda haber varias funciones con el mismo nombre pero con distintos identificativos. Universal Server determina qué función debe ser llamada durante la ejecución examinando su identificativo.

- Se consigue la encapsulación de objetos, permitiendo a sus diseñadores modificar su estructura interna o la de las funciones que operan sobre los objetos sin que se aprecie en las aplicaciones.

MÓDULOS DATABLADES

La gestión mediante DataBlade se realiza a través de módulos que desarrollan diferentes empresas además de la propia Informix.

En la actualidad ya existen en el mercado DataBlades para entornos financieros, cartografía, gestión documental, multimedia, comercio electrónico y Web y se están desarrollando otros. Estos DataBlades específicos son creados por empresas con experiencia en el sector concreto de desarrollo según el tipo de datos que se quiera modelizar.

Así empresas como Adobe Systems, Alta Vista, Eastman Kodak, EMC. MapInfo, Silicon Graphics, Sun Mycrosystems, Telecontar, Virage Technologies, entre otras muchas, están desarrollando nuevos DataBlades.

INFORMIX Universal Server gestiona a través de los mencionados DataBlades los siguientes tipos de información:

- GESTIÓN TEXTUAL

Con los módulos Text & Text Conversion DataBlades de INFORMIX-Universal Server, se pueden realizar aplicaciones como bancos de imágenes relacionadas por datos internos de las mismas o librerías de documentos.

Estos módulos posibilitan el desarrollo e implementación de sistemas documentales, permitiendo la gestión de nuevos tipos de datos y soportando funciones de conversión entre múltiples formatos.

Para el almacenamiento de los documentos se utiliza un método de indexación específico para tratamiento de texto, el llamado D-Tree, que agrupa las palabras por su raíz y permite al usuario el establecimiento de excepciones a la indexación que se propone. Se indexan todas las palabras de un documento, las frases y párrafos, lo cual permite la realización de búsquedas sobre cualquier palabra.

El almacenamiento se realiza en el formato del documento, aunque para la indexación se convierten en ASCII. En las operaciones de salida se permite la conversión a distintos formatos según la estación de trabajo del usuario final.

- GESTIÓN DE IMÁGENES

Las aplicaciones que pueden utilizar estos módulos en el SGBD pueden ir desde bancos de imágenes, catálogos, historiales clínicos, sistemas de reconocimiento de imágenes, documentación de atestados, librerías multimedia, etc.

Las imágenes se almacenan como si fueran datos simples y se reconocen hasta treinta formatos de imágenes. La gestión de las imágenes la realiza el motor de la base de datos a diferencia de la gestión realizada mediante BLOBs, donde la gestión se programaba en las aplicaciones, creciendo así su complejidad.

Mediante el Image Datablade, el desarrollador se abstrae de la gestión de las imágenes, centrándose en la aplicación.

- SERIES TEMPORALES

El TimeSeries DataBlade permite manejar información variable en el tiempo mediante la inclusión de dos tipos de datos, series temporales y calendarios. Dependiendo de la aplicación, las series pueden estar espaciadas segundos, minutos, horas días, semanas, según los requisitos previos del análisis de la aplicación.

Las series pueden manejar datos enteros, reales, monedas, información textual y espacial, y en general cualquier tipo de datos que se pueda representar en INFORMIX-Universal Server. El método de acceso para este tipo de objetos es la combinación de los métodos de acceso de la serie temporal y de su calendario. Estos datos pueden ser de utilidad para el análisis de la información del mercado de valores, análisis demográficos y sociológicos, farmacéuticos o médicos, etc.

DSA paraleliza las principales actividades de la base de datos, tanto las operaciones de manipulación, como las operaciones de administración de la base de datos

- INFORMACIÓN ESPACIAL DE 2 Y 3 DIMENSIONES

Las aplicaciones que utilizan el análisis y tratamiento de información espacial son muchas, las principales probablemente sean los sistemas de Información Geográfica (SIG) que en este sistema de gestión se pueden integrar con información corporativa.

Además podemos citar como aplicaciones típicas los sistemas CAD/CAM, los sistemas de control logístico de redes de distribución aérea, marítima o terreste, etc.

Los DataBlades 2D y 3D incorporan un conjunto de funciones destinadas a la gestión de información espacial en dos y tres dimensiones. Su objetivo primordial es facilitar el desarrollo e implementación de sistemas de información que se basen o que necesiten manejar coordenadas espaciales.

Incorporan tipos de datos para el soporte de figuras geométricas planas (polígonos regulares e irregulares) o en tres dimensiones (cubos, esferas,...) y más de mil funciones para la comparación, manipulación y creación de nuevos objetos a partir de los tipos de datos geométricos ya definidos.

Como ventaja de estos módulos, Informix señala que toda la programación se realiza utilizando SQL con lo que se simplifica el desarrollo de las aplicaciones, además se incorpora un método de acceso específico para este tipo de sistemas, el R-Tree, que permite direccionar la información según las coordenadas espaciales que ocupa, permitiendo así el acceso a la información sin necesidad de examinar a priori los objetos en los que está almacenada.

- INTERNET: LA WEB

Actualmente Internet está suponiendo una nueva manera de acceder, distribuir e intercambiar la información, pero además es una plataforma para la explotación de aplicaciones comerciales que requiere un acceso a fuentes de datos de muchos tipos.

Basándose en la importancia que esta forma de acceder a la información está adquiriendo día a día, el Web DataBlade está concebido como una

forma de proporcionar un conjunto de herramientas, funciones y prototipos que faciliten el desarrollo de aplicaciones sobre servidores Web teniendo como base el HTTP. Para ello se ha incorporado el concepto de Application Page (páginas de aplicación), en las que se pueden incluir ficheros HTML, consultas SQL, applets de Java, librerías C o cualquier programa que se necesite y que ya esté desarrollado.

La forma de realizar una consulta de usuario es distinta a la que estamos familiarizados dentro de Internet, ante una consulta de usuario, el Webdriver, que reside en memoria esperando cualquier petición de usuario, localiza una página de aplicación, la que corresponda al fichero HTML desde el que se produjo la consulta, después se generan los resultados de la consulta según la información almacenada en la base de datos a través de una función específica, que INFORMIX llama WebExplode.

De esta manera los servidores y las aplicaciones Web desarrolladas con INFORMIX-Universal Server son dinámicos, ya que la información de las páginas HTML y los enlaces a otros lugares del servidor Web o a otros servidores se realizan en tiempo de ejecución.

Como añadido en este módulo se incorpora un conjunto de funciones codificadas que permiten conocer en cada momento el uso que se está haciendo del servidor y de las aplicaciones Web que facilitan el desarrollo de aplicaciones y el conocimiento de las partes del servidor que deben ser rediseñadas si los usuarios tienen un único acceso.

OTROS MÓDULOS

En el momento de la redacción de este artículo existen más módulos DataBlade creados o en creación que cubren distintas tipos de objetos. Algunos de ellos son:

Datablades para aplicaciones Media:

- Reconocimiento de rostros, detección de escenas de vídeo.

-Vídeo.

- Reconocimiento de audio.

- Tecnología de marcas de agua.

- Gestión y búsqueda de imágenes.

- Encriptación de datos DES.

- Vídeo.

Para aplicaciones financieras y de Data Warehousing.

- Acceso y almacenamiento en medios ópticos.

Universal-Server de Informix es un motor de bases de datos avanzado, con arquitectura cliente-servidor concretada en la arquitectura dinámica escalable DSA, soportada por bases de datos relaciones y orientadas a objetos

Para aplicaciones de cartografía y espaciales:

- Tecnología de visualización.

- Espacial.

- Cartografía y geocodificación.

- Indexación espacial.

Para gestión documental:

- Gestión de documentos SGML.

- Búsqueda textual.

- Gestión y búsqueda de direcciones, empresas y nombres.

- Gestión de documentos LiveLink.

- Resumen y Agrupación de textos.

Para aplicaciones de comercio electrónico/Web:

- Generación de páginas Web dinámicas y soporte HTML.

- Gestión de objetos Web. Generación dinámica de "ofertas digitales".

- Integridad de datos.

- Búsqueda fuzzy.

- Series temporales.

- Sistema basado en Web de información para usuario.

EL PROGRAMA DE DESARROLLADORES

Para promocionar la creación de DataBlades, Informix ha diseñado un programa para desarrolladores de módulos que facilita la construcción de nuevos módulos de software. Este programa ofrece las herramientas (DataBlade Kit Developer), la formación y una serie de acciones conjuntas con las empresas de desarrollo para márketing.

El kit de desarrollo incluye herramientas basadas en IGUs que generan código C o SQL, herramientas de instalación, código fuente con ejemplos de Datablade, una guía de migración a Universal Server y documentación diversa.

En el programa participan empresas de distintos sectores de la informática: Adobe Systems Inc., en la creación de

módulos para recuperación y almacenamiento de ficheros PDF; Bulldog Group, para gestión de recursos multimedia; Comverse, reconocimiento de voz; Consistency Point Technol, acceso a datos en medios ópticos; Eastman Kodak, soporte de ficheros de imágenes FlashPix; Ecologic Corporation, tecnología de visualización; EDD, tecnología de integridad de datos; Excalibur Technologies, gestión/búsqueda textual; Macromedia Inc., gestión de ficheros Macromedia para el Web; Mathsoft Inc., análisis estadístico, Nichimen Data Systems Corp, tecnología de encriptación, Virage Technologies Inc. Gestión de imágenes basadas en contenidos, y otras.

LA ARQUITECTURA QUE SUBYACE

Unida a la metodología de la gestión de la información encuadrada en el paradigma de la orientación a objetos, nos encontramos con la plataforma hardware. El cambio de los tipos de datos que se utilizan en la concepción de la empresa a datos no estructurados y más complejos, también ha implicado un cambio en el soporte del software; de un lado se requieren sistemas escalables, es decir, que sean fácilmente adaptables a la gestión de mayores cantidades de información, por otro lado extensibles. La extensibilidad se consigue mediante los DataBlades, la arquitectura DSA (Dynamic Schalable Architecture) de Informix pretende ser la solución para la escalabilidad.

Como características de la arquitectura, Informix destaca las siguientes:

- Está diseñada para aprovechar las plataformas hardware en un entorno de sistemas abiertos, desde una máquina monoprocesador a un sistema de proceso paralelo, SMP (multiprocesador simétrico), sistemas cluster y MPP (procesadores masivamente paralelos, donde cada nodo es totalmente independiente pero a la vez todos cooperan para la ejecución en paralelo de la base de datos). DSA paraleliza las principales actividades de la base de datos, tanto las operaciones de manipulación (accesos, uniones, clasificaciones), como las operaciones de administración de la base de datos (creación de índices, copias de seguridad, recuperaciones, cargas y descargas).
- Escalabilidad máxima, en cuanto a que no se pone límites a la expansión del sistema, permitiendo que el rendimiento aumente a medida que sea necesario según la información, aprovechando las máquinas de proceso paralelo.
- Paralelismo interno en el núcleo del gestor. Los gestores DSA ejecutan múltiples operaciones en nodos distintos a la vez, además de poder descomponer las operaciones en tareas que serán distribuidas y ejecutadas independientemente en paralelo entre los nodos disponibles, soportando así el paralelismo vertical y el horizontal.
- Portabilidad de aplicaciones y datos, permitiendo aprovechar las ventajas del proceso paralelo sin modificar las aplicaciones. Se garantiza que los usuarios migren de un sistema a otro.

Pueden heredarse la estructura, las reglas (triggers en Universal Sever), los operadores, las funciones y las expresiones de agrupación, manteniendo así las ventajas de la OO

Para conseguir el máximo beneficio del procesamiento en paralelo, DSA se basa en la independencia de la plataforma y del hardware, estando diseñada para no depender de ningún aspecto concreto de ningún sistema operativo o plataforma.

Además se pretende optimizar la arquitectura proporcionando una total trasparencia de la aplicación. Se permite escalar a medida que se añaden usuarios al sistema utilizando procesos de servidor de la base de datos dinámicamente configurables que INFORMIX denomina procesadores virtuales. Éstos coordinan las consultas del usuario y las subtareas en paralelo (threads), de forma que un único procesador virtual pueda ser multiplexado entre varias threads. Las tareas pueden ser procesadas de forma concurrente, dividiendo las tareas en subtareas y aprovechando así las ventajas de un entorno multiprocesador.

CARACTERÍSTICAS

Las características que definen Universal Server, además de las mencionadas anteriormente, pretenden conseguir la escalabilidad y la flexibilidad.

Como características añadidas podemos mencionar:

1) Soporte de objetos de gran tamaño. Los objetos que exceden el tamaño que el servidor soporte, pueden almacenarse como ficheros externos o internos en la BD, pudiéndose recuperar en ambos casos en el supuesto de un posible fallo de la máquina.

2) Como utilidad del sistema, en máquinas multiprocesador, el cargador en paralelo gestiona tipos de datos abstractos y acelera el proceso de la carga de las BD que se añaden.

3) Para facilitar la extensibilidad de las herramientas del sistema de base

de datos, se utilizan los siguientes interfaces:

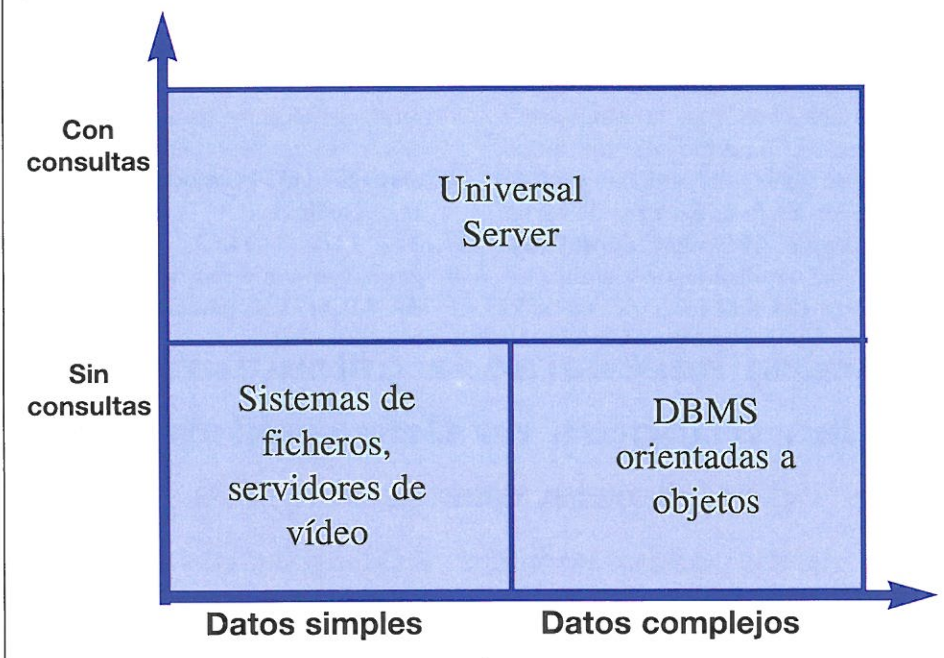
- Utilización de SQL embebido en C.
- Librerías de clases de C++, para soportar el desarrollo de aplicaciones en este lenguaje.
- Interfaz OLE para crear controles personalizados OLE (OCX) a la vez que los módulos DataBlade, utilizando la librería de C del nivel de llamadas para interactuar con Universal Server. Los OCX se distribuyen con módulos de ejecución de aplicaciones cliente y aparecen como iconos u opciones en la aplicación.
- Interfaz para Java. Una aplicación cliente de Java podrá descargar applets de Java de distintas redes; el applet se ejecutará entonces como la aplicación cliente. Además, se soporta la especificación de funciones de servidor en Java.

4) Se permite la creación de funciones de agregación (suma, media,...) con lo que se intenta facilitar la búsqueda de los datos. Estas funciones nuevas pueden ejecutarse en paralelo, en máquinas multiprocesador. La versión actual de SQL incluye algunas funciones como sum, average, etc. sin embargo no permite crear otras funciones nuevas.

5) El servidor soporta *triggers* sobre sentencias de selección (SELECT), permitiendo al SGBD ejecutar acciones si detecta que algún usuario se encuentra accediendo a determinados datos de la base de datos. Esta característica es importante fundamentalmente dentro de la WWW, ya que posibilita saber las normas de comportamiento de los usuarios e intentar determinar así la forma de navegación de los usuarios en visitas sucesivas.

6) El rendimiento de un SGDB viene determinado en gran parte por el método de acceso a los datos. US ha utilizado el método de indexación de árboles binarios (que ya utiliza Informix) para soportar tipos de datos abstractos, permitiendo un acceso rápido a aquellos que se puedan distribuir en una dimensión. Para datos de más de una dimensión se utiliza el índice de árbol R.

DSA está diseñada para aprovechar las plataformas hardware en un entorno de sistemas abiertos



7) Las conexiones para enlazar bases de datos Universal Server con otras se realizan mediante la Interfaz de métodos de acceso extensible (EAMI) que se ha implementado como una herramienta para gateways utilizada para crear gateways a sistemas distintos.

QUÉ ES UNIVERSAL-SERVER

Como resumen podría decirse que Universal-Server de Informix es un motor de bases de datos avanzado, con arquitectura cliente-servidor concretada en la arquitectura dinámica escalable DSA, soportada por bases de datos relaciones y orientadas a objetos para conseguir la gestión de contenidos no estructurados, destacando los contenidos multimedia dinámicos.

Las opiniones sobre el producto son variadas, pero desde Sólo Programadores pensamos que esta tecnología dará que hablar.

En la realización del artículo se han consultado fuentes pertenecientes a INFORMIX Software, Inc. entre las que destacan:

- -"Informix e Illustra se fusionan para crear Universal Server".
- -"Informix-Universal Server" de J.R. López, Product Marketing Manager.
- - "Informix Dynamic Scalable Architecture TM".
- - "Informix Datablade Technology".
- - Notas de prensa de Informix.

Además de otras como contraste, de interés para el autor.

TCL/TK. HERRAMIENTAS ABIERTAS

José Antonio Collar Ruano

El abanico de sistemas hard\soft sobre los que podemos desarrollar con diversas distribuciones de Tcl/Tk es amplísimo. Las diferencias entre las bibliotecas básicas en la inmensa mayoría de los casos son apenas perceptibles, con lo que la portabilidad de rutinas básicas está garantizada. Veamos una relación (no exhaustiva) de estas plataformas:

La especialización de los lenguajes y las herramientas ha roto definitivamente el concepto tradicional de desarrollo cerrado, dando paso a un modelo cooperativo y jerarquizado

- **Unix.** Como es lógico, la lista más poblada. Y con la mejor portabilidad entre sus miembros.

  -Apollo / BSD/SYSV
  -Cray / Unicos
  -DECstations / Ultrix
  -DEC VAX / Ultrix ó BSD
  -DEC Alpha / OSF/1
  -Encore 91's / UMAX V
  -HP / HP-UX
  -i386 / SCO Unix, Xenix, Bell-Tech, netbsd, freebsd, BSDI, Linux
  -IBM RS6000 / AIX 3.x, IBM ES/9000 , AIX/ESA.
  -Mac / A/UX
  -Sequent Symmetry / Dynix, OSF/1
  -Silicon Graphics / IRIX
  -Sun 3 y 4 / SunOS 4 y SunOS 5 (Solaris 1.x y 2.x)
  -Varios / System V.4

- **Intel / D.O.S. / Windows.** Con algunas especificidades más acusadas.

  -MS-D.O.S.
  -MS Windows 3.x
  -MS Windows 95
  -MS Windows NT
  -MS Windows DLL, Windows Sockets
  -NetWare
  -OS/2 2.x / C Set++
  -OS/2 Presentation Manager
  -QNX

- **Mac.**

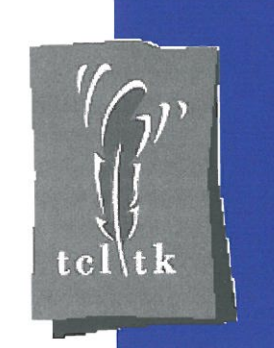
  -MacOS

- **VMS.**

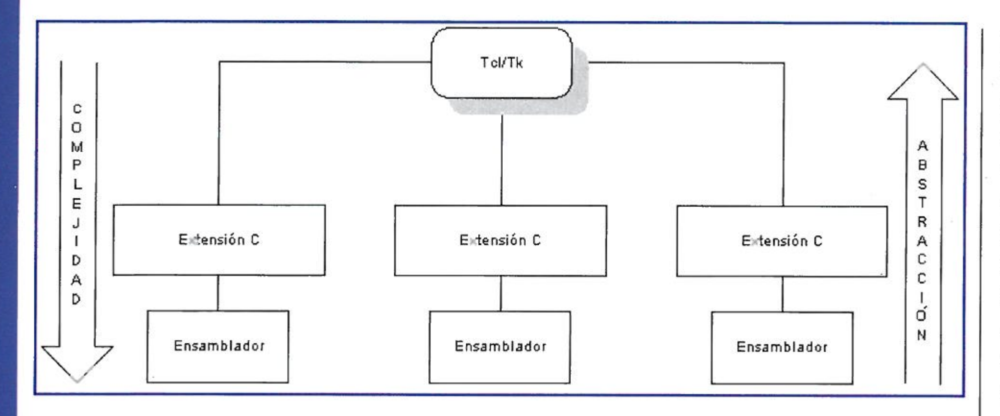
  -Alpha / OpenVMS T6.1.
  -VMS 5.5

- **Otros.**

  -AmigaOS
  -Amiga / NetBSD1.0b2



Si en el ámbito de los sistemas el concepto de "abierto" se ha impuesto de modo implacable, en el campo de la programación no podía ser menos. Soportadas sobre múltiples plataformas e integradas de una forma sencilla y directa con otros lenguajes de programación, las herramientas que componen Tcl/Tk son uno de sus paradigmas.



-Apple IIgs
-BeOS Dr7
-GNO 1.1/GSOS
-NeXT
-OS-9
-VxWorks

Esta lista se amplía continuamente, aunque por supuesto y como regla general, el soporte y la profundidad de las diferentes posibilidades mostradas corren parejos a la implantación de la plataforma en cuestión.

INTEGRACIÓN CON LENGUAJES

Recordemos que Tcl fue ideado con la filosofía de que deberíamos realmente usar dos o más lenguajes cuando diseñamos grandes sistemas software. Uno para manipular estructuras internas de datos complejas, o cuando la optimización del rendimiento es vital, y otro, como Tcl, para escribir pequeños scripts que aten entre sí todas las piezas. Esto se ilustra con la figura 1, donde vemos que los lenguajes tienden a especializarse en estratos lógicos, para llevar a cabo las tareas que le son más cercanas. No es extraño pues que la integración con otros lenguajes, y como no, muy especialmente con C, sea una de sus características más notables. Además de C, tenemos enlaces disponibles, entre otros, con C++ , Perl, Modula 3, LISP y Ada.

Esta filosofía cobra sentido pleno con el desarrollo sobre Tk, ya que es precisamente este tipo de herramientas de diseño de entorno gráfico la que tiende más hoy en día a la abstracción de métodos, pero basándose en funciones no nativas de bajo nivel para procedimientos de datos complejos.

EXTENSIONES EN C

Estudiaremos el escenario más potente por el momento, que corresponde a la extensión de Tcl/Tk con rutinas en lenguaje C.

Todos los comandos Tcl basados en C se han de llamar con los cuatro argumentos siguientes:

- Un puntero a los datos cliente.
- Un puntero al intérprete.
- Un contador de argumentos.
- Un puntero a un array de cadenas de caracteres con los argumentos Tcl para el comando.

Lo que sigue es un ejemplo sencillo de una extensión Tcl en C:

```
#include "tcl.h"

int App_EchoCmd(clientData, interp,
argc, argv)
   void *clientData;
   Tcl_Interp *interp;
   int argc;
   char **argv;
{
   int i;

   for (i = 1; i < argc; i++) {
      printf("%s",argv[i]);
      if (i < argc - 1) printf(" ");
   }
   printf("\n");
   return TCL_OK;
}
```

El puntero al intérprete es la clave para un intérprete. Es devuelto por *Tcl_CreateInterp* y se utiliza de un modo muy común dentro de Tcl, así como en todas las extensiones que generemos en C. La estructura de datos apuntada por el puntero de intérprete, así como todas las estructuras subordinadas que surgen desde ella, conforman un intérprete Tcl, que incluye todos los procedimientos definidos actualmente, junto con todos los comandos, variables y arrays. Alberga así mismo el estado de ejecución del intérprete en cuestión. El contador de argumentos y el puntero al array de argumentos en formato de cadenas de texto es manejado por el código C del mismo modo que lo haría al escribir un función *main* en C. El contador y el puntero funcionan igual que en una llamada a una función *main*; los punteros a los argumentos de la función están contenidos en el array *argv*. Al igual que en una función *main* en C, el primer argumento (*argv[0]*) es el nombre con el que se llamó a la rutina (en un *main*, el nombre con el que el programa fue invocado).

En el ejemplo anterior, todos los argumentos son devueltos con un espacio de separación mediante un bucle a través del array *argv*, desde uno hasta el valor del contador *argc*.

VALIDACIÓN DE ARGUMENTOS

Todos los argumentos desde una llamada Tcl a una extensión C de Tcl se pasan como strings. Si la rutina C espera argumentos numéricos la rutina deberá primero convertirlos usando las funciones *Tcl_GetInt* ó *Tcl_GetDouble*. También pueden utilizarse las funciones extendidas *Tcl_GetLong* ó *Tcl_GetUnsigned*, u otro método distinto que imaginemos. Del mismo, para convertir valores booleanos deberemos utilizar *Tcl_GetBoolean*. Estas rutinas depositan automáticamente un mensaje de error apropiado en el buffer de resultados del intérprete Tcl y devuelven *TCL_ERROR*, si es que se produce un error de conversión.

Ocurre algo similar si el programa produce un resultado numérico, ya que deberá devolver una cadena equivalente al valor numérico. Una forma simple y habitual de hacer esto es algo parecido a esto:

```
sprintf(interp->result, "%ld", result);
```

Sólo cuando tratamos con resultados relativamente cortos es una buena idea

escribir los resultador directamente dentro del buffer de resultado del intérprete. Tcl tiene una función, *Tcl_SetResult*, que proporciona la capacidad de que las extensiones C que programemos devuelvan cadenas muy extensas a Tcl, con la peculiaridad de indicarle al intérprete si él tiene la posesión de la cadena (con lo que Tcl deberá borrarla cuando termine con ella), si la cadena puede ser cambiada o sobreescrita pronto (con lo que Tcl deberá hacer una copia de la cadena), o si la cadena no cambiará (con lo que Tcl puede usarla directamente y no ocuparse más de ella). Comprender la forma en que los resultados se devuelven a Tcl es esencial para escribir correctamente extensiones en C.

Los comandos sofistificados deberían verificar sus argumentos siempre que fuera posible, examinando el contador de argumentos, comprobando que los campos numéricos son realmente numéricos, que los valores están dentro de rango (cuando se conozcan sus rangos), etc.

Tcl está diseñado en principio de un modo "blindado", en el sentido de que ningún programa Tcl debería ser capaz de causar la corrupción del núcleo Tcl. Se ha de ser cuidadoso en la validación de los argumentos en la integración mediante extensiones C.

CONCLUSIONES

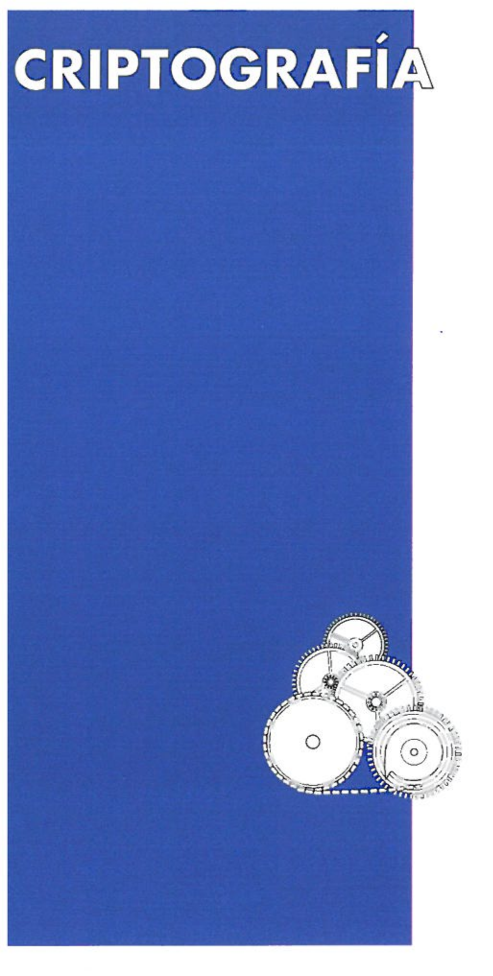
Como hemos podido ver, el conjunto de herramientas Tcl/Tk se puede utilizar en tantos entornos y junto a tantas aplicaciones adjuntas, que resulta idóneo para funcionar como esa "argamasa" de unión entre diversos módulos para la que fue diseñado. Gracias a él podremos mantener el esqueleto de nuestros sistemas software de un modo eficiente y poderoso, incluyendo diversas plataformas.

Las extensiones en C son un potente ejemplo de la convivencia entre Tcl y otros lenguajes de programación

Podemos por ejemplo mantener un mismo esquema basado en Tcl para un proyecto software que involucre múltiples sistemas, y con ellos elementos diferenciales de bajo nivel.

De este modo nos bastará incidir menos en los ciclos lógicos del sistema y centrar el esfuerzo en la resolución de esas diferencias básicas.

Debemos tener en cuenta del mismo modo que estas características vienen dadas por la propia filosofía del lenguaje, ya que es relativamente sencillo abordar la tarea de construir la biblioteca básica para una nueva plataforma, o abrirla a rutinas externas, con lo que de hecho las posibilidades se hacen ilimitadas.



SI QUEREMOS QUE NUESTROS DATOS SEAN SECRETOS

F.G. Aviles

Actualmente es posible enviar y recibir mensajes con alguna intimidad e integridad con el uso de la informática. No obstante, la seguridad en los mensajes y ficheros enviados a través de redes públicas telefónicas -dado el uso general de Internet- no es sólo un problema tecnológico, sino también un reto cultural y social.

¿POR QUÉ DEBO CIFRAR MI CORREO ELECTRÓNICO?

Los expertos dicen que por la misma razón que cerramos las cartas y no se dejan abiertas.

Ahora mismo, cualquier mensaje en Internet o en FIDO pasa por bastantes nodos. Y seguro que hay mucho curioso suelto y gente con malas intenciones que puede y quiere abrir nuestras cartas para apoderarse de nuestro trabajo y nuestro conocimiento. También hay quien dice que Encriptar es divertirse.

Amparándose en el derecho a la intimidad de las comunicaciones, hay grupos delictivos que se aprovechan para intercambiar la información para realizar sus delitos -narcotráfico, terrorismo, trafico de divisas, etc ...- En España, como es conocido, no existe ningún tipo de limitación en el cifrado de la transmisión de datos. En Francia, existe una ley que sólo permite el empleo del cifrado con una orden especial de las autoridades que, además, indican el tipo de cifrado a emplear -de esta forma, se considera ya un delito el simple hecho de cifrar una información sin autorización-. Es la filosofia basada en que al delincuente tal vez no le cojan por su delito, pero sí por la ocultación de lo que está transmitiendo para hacer el delito.

En EE.UU., existe la iniciativa Clinton sobre el chip Clipper, que es un sistema que permite, por un lado, proteger la información a transmitir mediante un sistema robusto y, por otro, ante un delito y con las correspondientes garantías legales, permitir a las autoridades descifrar fácilmente la transmisión fraudulenta.

Encontrar un equilibrio es difícil y normalmente en el término medio se encuentra la virtud. Netcape permite el uso de criptografía. (Ver FIGURA 1).

PRETTY GOOD PRIVACY

En la actualidad existen diversos programas para criptografía, como ejemplo vamos a ocuparnos del PGP, el Pretty Good Privacy. El autor de este sistema de cifrado fue el estadounidense Phillip Zimmermann que estuvo encausado varios años por haber puesto este sistema de cifrado en la red Internet.

En Estados Unidos todavía se sigue considerando los sistemas de cifrado como productos estratégicos de interés para la Defensa Nacional de dicho país -están en la misma categoría que la munición de guerra- y, por tanto, tienen grandes restricciones para la exportación. De hecho, pueden exportar sistemas de cifrado en versiones muy disminuidas -por ejemplo, con longitudes de clave de 40 bits que son fácilmente "rompibles" ante un ataque de criptoanálisis con fuerza-. No obstante, como la mayoría de estos algoritmos son públicos, existen versiones completas desarrolladas por empresas europeas que los comercializan tanto en sus versiones hardware como software.

Respecto al PGP, es un sistema mixto de intercambio de claves públicas por sesión, que sirven para el

La criptografía o cifrado es uno de los medios usados desde la antigüedad para conseguir seguridad y confidencialidad en las comunicaciones. Ya un autor clásico del siglo XIX convertía un mensaje cifrado que empezaba con la serie de letras INYJSGDGGXPCh, con la clave repetitiva númerica 432513 en "El verdadero a...."

cifrado simétrico de la información. Es decir, se cifran los datos a transmitir con la clave pública del destinatario que sólo puede ser descifrado por éste con su clave privada (secreta) -el algoritmo está basado en propiedades matemáticas como la utilización de logaritmos discretos o la factorización de grandes números enteros primos.

Si se quiere recibir un mensaje cifrado, el emisor debe conocer la clave pública para poder mandar un mensaje confidencial que sólo se podrá descifrar con la clave privada. Por tanto, existen unos sitios públicos que dan el servicio de facilitar las claves públicas de los usuarios que empleen el PGP. En España, uno es el CERT de la red universitaria/científica RedIris. Aunque, evidentemente, existen otros muchos sitios donde depositar la clave pública -pues la privada se debe mantener en secreto-. Nadie es obligado a depositar la clave pública en ese servidor, pero si no es depositada en algún sitio de acceso público, nadie podrá enviar un mensaje cifrado al dueño de la clave.

Los aspectos legales de los algoritmos de encriptación están empezando a adquirir complejidad, en España no existe ningún tipo de legislación sobre el uso de los sistemas criptográficos, al menos a nivel privado. Actualmente hay un grupo de especialistas en la Unión Europea, llamado SOGIS, que está intentando normalizar una legislación sobre el uso de la criptografía en los países miembros de la UE. Esta legislación tendrá una aplicación directa en las medidas de protección que la LORTAD obliga a tomar en el tratamiento de datos personales que los organismos/empresas públicos y privados tienen en España.

El PGP emplea el algoritmo IDEA como sistema de cifrado y utiliza el sistema RSA para el intercambio de claves de sesión. El IDEA (*International Data Encryption Algorithm)* fue desarrollado por la empresa ETH Zurich en 1.992, como sistema de cifrado simétrico con claves de 128 bits de longitud, cuya versión no comercial es de libre uso y público conocimiento.

El par de claves pública/privada son complementarias, es una función exponencial irreversible. Es decir, lo que cifras con la clave pública, sólo y únicamente sólo, se puede descifrar con la clave privada y, viceversa, sólo y únicamente sólo, se puede descifrar con la clave pública lo que hayas cifrado con tu clave privada. Como sistema de cifrado asimétrico o de clave pública se puede depositar la clave pública en cualquier servidor para este fin, tal como *http://www.rediris.es/cert/key-server/*

Como resumen de los comandos del PGP consúltese el cuadro 1.

CÓMO SE PUEDE OBTENER EL PROGRAMA PGP

La versión comercial se obtenía de una compañía llamada ViaCrypt -de Phoenix, Arizona - y actualmente de PGP - Pretty Good Privacy,Inc - que en julio del 96 compró a la anterior. Aunque están sacando nuevos productos para posibilitar el cifrado del e-mail y el teléfono vía Internet, las ventas del PGP comercial sólo serán para uso interno en EE.UU. de acuerdo con el Departamento de Justicia estadounidense. Las versiones de uso común en Europa se encuentran en muchos servidores de Internet, como el conocido *ftp.funet.fi* y en algunas BBS, algunos sitios son:
http://www.rediris.es/cert/key-server/
Claves públicas
http://www.encomix.es/pgp/pgp-doc1.html. Es un verdadero manual del PGP en español

CONSEJOS DE USO DE CIFRADO Y PERMISOS

Normalmente en Internet todavía hay bastante libertad en cuanto al uso de

CUADRO 1.

Para encriptar un fichero normal con la clave pública del destinatario:
pgp -e fnormal identificador_des [otros identificadores]
Para firmar un fichero normal con su clave secreta:
pgp -s fnormal [-u su_identificador]
Para firmar un fichero normal con su clave secreta y después encriptarlo con la clave pública del destinatario:
pgp -es fnormal identificador_des [otros id.] [-u su_identificador]
Para encriptar un fichero normal sólo con criptografía convencional, escriba:
pgp -c fnormal
Para desencriptar un fichero, o comprobar la integridad de la firma en un fichero firmado:
pgp fcifrado [-o fnormal]
Para desencriptar un fichero con armadura ASCII en varias partes: grabe todas las partes por orden en un fichero con extensión *.asc* y escriba:
pgp farmadura [-o fnormal]
—- Funciones para la gestión de claves —–
Para generar su propio par único de claves pública/secreta:
pgp -kg
Para añadir el contenido de un fichero de claves públicas o secretas al anillo correspondiente:
pgp -ka fdclaves [anillo]
Para extraer (copiar) una clave del anillo de claves públicas o secretas:
pgp -kx identificador fdclaves [anillo]
o: pgp -kxa identificador fdclaves [anillo]
Para visualizar el contenido del anillo de claves públicas:
pgp -kv[v] [identificador] [anillo]
Para ver la "huella dactilar" de una clave pública, y poder verificarla por teléfono con su dueño/a:
pgp -kvc [identificador] [anillo]
Para visualizar el contenido y comprobar las firmas de certificación en el anillo de claves públicas:
pgp -kc [identificador] [anillo]
Para modificar el identificador o contraseña de la clave secreta:
pgp -ke identificador [anillo]
Para modificar los parámetros de confianza de una clave pública:
pgp -ke identificador [anillo]
Para suprimir una clave, o s_lo un identificador, del anillo de claves públicas:
pgp -kr identificador [anillo]
Para firmar la clave pública de alguien en el anillo de claves públicas:
pgp -ks identificador_des [-u su_identificador] [anillo]
Para desencriptar un mensaje y dejar su firma intacta:
pgp -d fcifrado
Para crear un certificado de firma separado del documento:
pgp -sb ftexto [-u su_identificador]
Para separar un certificado de firma del mensaje firmado:
pgp -b fcifrado
Para visualizar el texto desencriptado en la pantalla (como con la orden "more" {más} de Unix/MSDOS), sin grabarlo en un fichero, utilice la opción -m (more) al desencriptar:
pgp -m fcifrado

programas de cifrado/descifrado de datos, de todas maneras conviene preguntar al *postmaster o websmaster* de nuestro proveedor Internet. Aparte del correo electrónico con USA en el cual no podemos usar la versión americana del PGP, sino la internacional.

En cuanto a nuestra BBS preferida y la red FIDO debemos tener en cuenta tres cosas:

1) Que nuestro Sysop permita el uso del cifrado de datos.

2) El Sysop de nuestro remitente/recibidor también lo permita.

3) Los Sysops de los nodos donde se ruta nuestro correo también lo permitan.

Y los consejos generales de cualquier comunicación:

- Como no dar nuestra clave secreta privada a nadie.
- No abusar del uso de cifrado por operatividad, sólo cifrar lo más secreto.
- No ser fiadores de gente desconocida o sospechosa de actividades ilegales.
- Confirmar la primera vez por teléfono con el destinatario la buena recepción de los mensajes.
- Aprender de los expertos y realizar pruebas.

GLOSARIO DE TÉRMINOS DE CIFRADO/CRIPTOGRAFÍA

Cifrado - Transformación de una información (Texto Claro) en otra ininteligible (Texto Cifrado) según un procedimiento y usando una clave determinados, que pretende que sólo quien conozca dichos procedimientos y clave pueda acceder a la información original. Es un mecanismo de seguridad.

Clipper - Un chip desarrollado por el Gobierno de USA que sería usado en todas las comunicaciones cifradas.

Criptología - Ciencia que estudia la ocultación, disimulación o cifrado de la información, asl como el diseño de sistemas que realicen dichas funciones, e inversamente la obtención de la información protegida. Comprende el Cifrado y el Criptoanálisis.

Criptoanálisis - Pasos y operaciones orientadas a transformar un criptograma en el texto claro original pero sin conocer inicialmente el sistema de cifrado utilizado y/o la clave.

DES (Data Encryption Standard) - Norma de encriptación de datos desarrollada por el Gobierno de Estados Unidos.

EFF (Electronic Frontier Foundation) - Fundación fundada en julio de 1990, para asegurar la libertad de expresión en los medios digitales, con un particular énfasis en aplicar los principales derechos contenidos en la Constitución U.S.A, etc en las comunicaciones basadas en ordenador.Internet: eff@eff.org.

IDEA (International Data Encryption Algorithm) - Algoritmo desarrollado en Suiza y licenciado para uso no comercial en PGP.- Para más detalles VER apartado sobre él dentro de este artículo -

ITAR (International Traffic in Arms Regulations) - Leyes de Estados Unidos que cubren la exportación de armas y su tecnología. Incluyen las técnicas y productos de cifrado de datos.

NSA (National Security Agency) - Agencia oficial del gobierno U.S.A que vela por la seguridad en las comunicaciones. Es la que se encarga con sus grandes ordenadores del cifrado/descifrado.

RSA (Rivest-Shamir-Adleman) - Método de cifrado de la clave pública usado en el programa PGP. El acrónimo RSA corresponde a las iniciales de los desarrolladores de su algoritmo.

EL ALGORITMO DE CIFRADO IDEA

Debido a la previsible caducidad del DES, los más prestigiosos criptólogos de todo el mundo trabajan en el desarrollo de algoritmos que le sustituyan con la mayor compatibilidad posible. Este es el caso del IDEA, que ha sido diseñado con el propósito de resistir a ciertos ataques a los que el DES es vulnerable, en especial a los denominados como "criptoanálisis diferenciales", y ofrecer un tamaño de la clave suficientemente seguro.

El International Data Encryption Algorithm fue inventado en Suiza por el doctorando chino Xuejia Lai y James Massey cuando ambos trabajaban en el Institute for Signal and Information Processing del Instituto Federal de Tecnología Suizo, en Zurich. Se patentó en Noviembre de 1991.

El titular de la patente (naturalmente sólo en los pocos países en los que pueden protegerse los derechos industriales de un algoritmo matemático, entre los que no se encuentra España, ni la Unión Europea o Japón) es la empresa de telecomunicaciones suiza Ascom Tech AG, la cual permite el uso gratuito del programa para fines no comerciales. Los términos de la licencia para fines comerciales parecen bastante razonables incluso para pequeñas y medianas empresas de tecnología básica para informática y comunicaciones.

El IDEA es un cifrador de bloques que opera sobre secuencias de 64 bits en cada ciclo, es decir, que a partir de 64 bits de texto en claro, obtiene otros 64 bits de texto cifrado, según determina una clave de 128 bits de tamaño, lo cual es más que suficiente con la actual potencia de cálculo para garantizar la invulnerabilidad frente a ataques exhaustivos.

Tal y como sus autores exponen en su propuesta ["A proposal for a new block encryption standard", X. Lai y J. Massey, Advances in Cryptology - EUROCYPT`90], el algoritmo de cifrado está basado en conceptos de diseño que mezclan operaciones procedentes de grupos algebraicos diferentes, persiguiendo las mejores propiedades contra criptoanálisis.

Al ser un algoritmo de clave secreta, el procedimiento y la clave de descifrado es el mismo que el utilizado para el cifrado. Sin embargo, el algoritmo IDEA es perfectamente combinable con otros asimétricos, como el RSA, y funciones resumen, como el MD-5 de RSA Inc.

La confusión necesaria en todo algoritmo criptográfico, entendida como caótica independencia del texto cifrado respecto al claro, se consigue en el IDEA mediante tres grupos diferentes de operaciones sobre parejas de subbloques de 16 bits, y la estructura del cifrador se ha elegido de forma que proporcione una óptima difusión, es decir, que un sólo bit cambiado en el texto claro provoca que sea diferente todo el texto cifrado, lo que facilita la divergencia en sucesivas reiteraciones. Por otro lado, los criterios de diseño han pretendido que sea fácil su construcción tanto en dispositivos físicos como lógicos.

Después de dar los detalles del diseño, es posible formarse una sinopsis aproximada de lo considerablemente

TE BUSCAMOS...

SI TIENES AMPLIOS CONOCIMIENTOS O EXPERIENCIA EN ALGUNO DE ESTOS CAMPOS

- Programación en Visual Basic, Visual C++, Delphi, HTML, JAVA, JAVAScript, CGI
(Ref. PROG)

- Lenguajes multimedia como Director, Authorware, Toolbook
(Ref. MULT)

- Dominio de sistemas operativos DOS, Windows 95, Windows NT, UNIX, Linux, OS2.
(Ref. S.O)

- Redes, Conectividad y Sistemas abiertos
(Ref. RED)

- Diseño y autoedición en CorelDraw, Photoshop, FreeHand, QuarkXPress, Page Maker
(Ref. DIS)

- Infografía con 3D Studio, LightWave
(Ref. INF)

- Internet, Navegación, Videoconferencia, ActiveX, Plug-Ins, Shockwave, RDSI
(Ref. NET)

- Nuevas tecnologías, MMX, USB, DVD, Telefonía móvil
(Ref. TECN)

- Instalación, reparación y mantenimiento de software y hardware
(Ref. INST)

- Hardware: Tarjetas gráficas, Modems, Cámaras digitales, Scaners, Tarjetas de sonido
(Ref. HARD)

- Inteligencia Artificial, Sistemas Expertos, Redes neuronales, Robótica
(Ref. SE)

COLABORA CON NOSOTROS

Envíanos un currículum vitae con la referencia correspondiente a la siguiente dirección:
TOWER COMMUNICATIONS - APARTADO DE CORREOS 54.283 - 28080 MADRID

robusto que es este algoritmo frente a los principales ataques criptoanalíticos conocidos.

UNA DESCRIPCIÓN ESQUEMÁTICA DEL IDEA

En cada paso elemental por el que pasa el texto de 64 bits, éste sufre las siguientes alteraciones para dar el cifrado:

1ª División del bloque de 64 bits de texto en cuatro subbloques de 16 bits: X1, X2, X3 y X4, y de los 128 de la clave en 8: Z1 ... Z8. Estos últimos son intencionadamente más complicados, y se tratarán aparte.

2ª Multiplicación de X1 por el primer subbloque de la clave Z1

3ª A ello se añade X2 y el segundo subbloque de la clave Z2

4ª A lo que se añade X3 y el segundo subbloque de la clave Z3

5ª Lo que se multiplica por X4 y por el cuarto subbloque de la clave

6ª Se hace una operación de or exclusivo (XOR) entre el bloque resultado de las operaciones segunda y cuarta

7ª XOR entre el resultado de la tercera y la quinta operación

8ª Se multiplican los resultados de la sexta operación con el quinto subbloque de la clave, Z5

9ª Se suma el resultado de las operaciones sexta y séptima

10ª Se multiplica el resultado de la novena operación con el sexto subbloque de la clave, Z6

11ª Se suman los resultados de la octava y la décima operación

12ª XOR del resultado de la segunda y la décima

13ª XOR del resultado de la cuarta y la décima

14ª XOR del resultado de la tercera y la undécima

15ª XOR del resultado de la quinta y la undécima

De esta forma se consigue el texto cifrado, juntando los cuatro bloques resultantes de las cuatro últimas operaciones.

Evidentemente, la seguridad del cifrador aumenta cuando el texto pasa varias veces por el algoritmo, y el IDEA lo hace 8 veces. Para retroalimentarlo, basta con permutar los bloques segundo y tercero de la salida, y hacerlos entrar de nuevo, en todas las etapas menos en la última. Después de la octava ronda, hay una transformación final, que consiste en los cuatro pasos siguientes:

1º Multiplicar X1 por el primer subbloque de la clave

2º Añadir X2 al segundo bloque de la clave

3º Añadir X3 al tercer subbloque de la clave

4º Multiplicar X4 por el cuarto subbloque de la clave

Finalmente, se concatenan estos últimos cuatro bloques para conseguir el texto cifrado de salida. La creación de los subbloques de la clave, en el 1er paso del algoritmo, consiste en dividir los 128 bits de dicha clave en ocho subbloques de los cuales sólo seis se utilizan en la primera vuelta, reservando dos para la segunda. Tras esto, la clave se hace rotar 25 bits a la izquierda (mediante la función de desplazamiento de los bits en la palabra que proporciona el operador "<<" del lenguaje de programación C), y de nuevo se divide la clave en ocho subclaves. Las cuatro primeras se utilizan en la segunda etapa del algoritmo, y las cuatro últimas en la tercera. La clave se rota otros 25 bits a la izquierda para conseguir las ocho siguientes subclaves, y así sucesivamente.

En esencia, para lograr la simetría computacional necesaria para la clave secreta, se aprovecha la siguiente igualdad: 216x215 mod (216 + 1) = 215 + 1

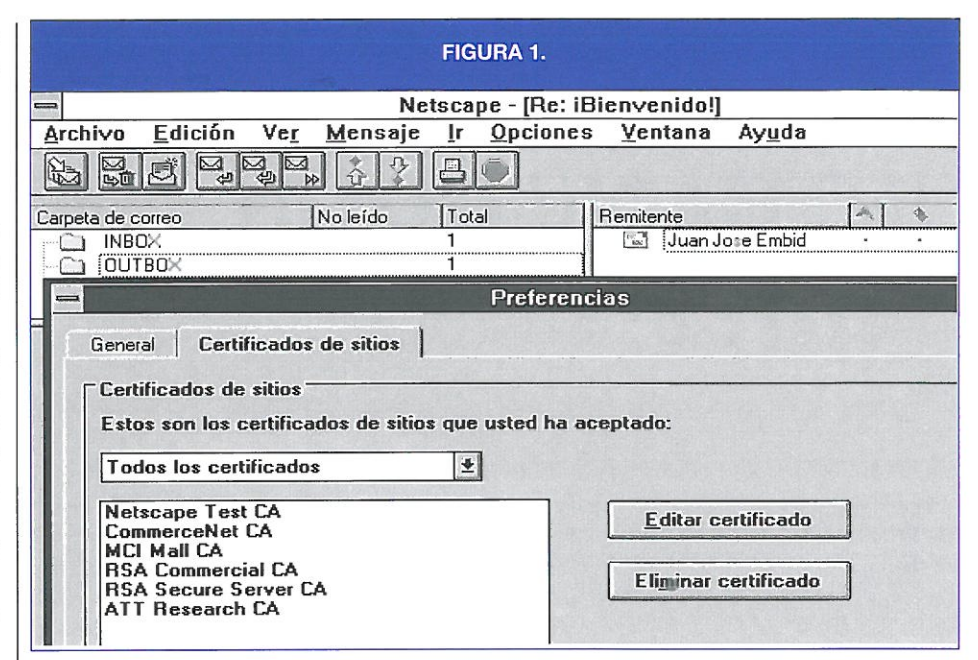


FIGURA 1.

RENDIMIENTO DEL CIFRADOR IDEA

Las actuales implementaciones de este algoritmo son de dos a cuatro veces más rápidas que las equivalentes del DES (Data Encryption Standard). En un antiguo Intel 386 a 33 Mhz es capaz de ofrecer una velocidad de cifrado de unos 880 Kbps, y en un VAX 9000 de unas cuatro veces más. Un chip VLSI desarrollado en el ETH de Zurich, formado por 251.000 transistores en una pastilla de 107.9 mm de lado, cifra con el algoritmo IDEA a una velocidad de 177 Mbits/sec. con un reloj de 25 Mhz.

Los interesados deben saber que el fabricante proporciona en disquete la versión 1.0 para Unix, que incluye un manual completo en inglés del criptosistema IDEA según ASCOM.

BIBLIOGRAFÍA

- Seguridad y Protección de la Información, de J.L.Morant Ramon, A. Ribagorda Garnacho y J.Sancho Rodríguez. Editorial Centro de Estudios Ramón Areces, S.A. Colección Informática, 1.994
- Códigos Secretos, de Andrea Sgarro, Ediciones Pirámide, S.A., Madrid, 1.990.
- Applied Cryptography, de Bruce Schneier, Editorial John Wiley & Sons, Inc 1.994 y segunda edición de 1.996 (bastante más ampliada y mejorada).

CREACIÓN DE UNA INTRANET CON LINUX

Carlos Álvaro

La aplicación a una red privada de la filosofía Internet, de su estructura y sus servicios, ya sea en una red corporativa de una empresa o de cualquier otra organización, es lo que se conoce como una "intranet".

La tecnología Internet es fácil de aplicar a una red, fácil de utilizar, poco costosa y con prestaciones de gran calidad. No es extraño entonces su gran auge en todos los entornos, últimamente y de forma especialmente notable en el mundo de la empresa que, cada vez más, demandan profesionales o servicios de otras empresas relacionados con el mundo Internet y con las internets.

No es ahora nuestro objetivo el de definir exhaustivamente lo que es una intranet, sus servicios o sus posibilidades y ventajas. En este número de "Sólo programadores" pretendemos hacer una introducción a lo que, a lo largo de los próximos meses nos servirá para montar nuestra propia intranet. Para ello, utilizaremos exclusivamente productos de los llamados de "libre distribución", (aunque este término, en muchos casos, se utiliza erróneamente), con lo que los costes se reducirán al equipo o equipos que queramos utilizar, la circuitería, el cableado, el uso de una conexión a la Internet necesario para la obtención del software y... nuestro tiempo y buena disposición.

Sí es conveniente, sin embargo, hacer algunas precisiones relacionadas con el concepto de las intranets, sobre todo en lo que implican los requisitos a cumplir para poder denominar "intranet" a nuestra red.

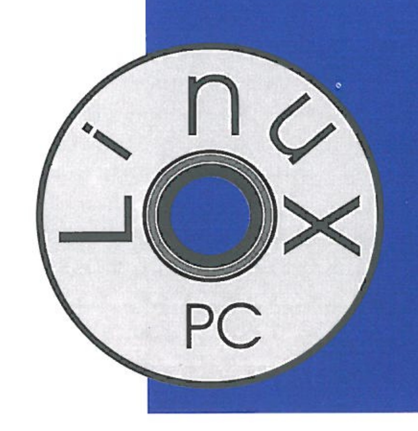
¿Qué requisitos debe cumplir una red para que sea considerada una "intranet"? Pues principalmente dos: el primero, que el diseño de su arquitectura sea similar al de la Internet, es decir, que utilice los protocolos de comunicaciones estandarizados para la Internet (TCP/IP), ofreciendo a través de ellos los servicios propios del ámbito Internet y, el segundo, que se trate de una red privada, es decir, que no sea visible al resto de la comunidad Internet.

Precisando un poco más este segundo requisito, diremos que una estación de trabajo conectada a una intranet no es accesible desde el exterior, sin embargo, ella sí que tiene acceso al conjunto de los servidores públicos de la Internet. Estos últimos pueden, a su vez, ser parte de otras intranets, por lo que son los enlaces lógicos entre ellas y el resto de la Internet.

Durante los próximos números de "Sólo Programadores", nuestra construcción de una intranet se desarrollará en tres vertientes fundamentales que se llevarán a cabo en paralelo.

La primera de esas vertientes viene dada por los problemas y peculiaridades de las conexiones TCP/IP. Sin entrar en demasiadas cuestiones a bajo nivel, pretendemos dar una visión general del funcionamiento de esta familia de protocolos, en un principio, de la configuración para Linux, aunque tenemos previsto hablar de otros sistemas operativos del tipo de OS/2, Windows 95, Windows NT, etc.

La segunda nos llega asociada a los servicios que vamos a ofrecer. Entraremos en la mayor o menor complejidad de la instalación y configuración de servidores, de Web, FTP, etc, y los enlaces entre estos programas ser-



La tecnología Internet ha abierto las puertas a una concepción nueva de la sociedad y, en ella, de la empresa, desde los métodos de investigación hasta la forma de comunicación. Durante unos meses, crearemos nuestra intranet y la convertiremos en un mundo propio.

vidor con sus programas cliente en la estación de trabajo.

La tercera, que es un poco posterior, se refiere a los elementos de control y mejora del rendimiento que habitualmente se instalan en las intranets: monitorización del tráfico, firewalls y proxies.

Será nuestra necesidad de realizar una tarea la que nos llevará a hacer un repaso por las cuestiones asociadas a la misma. Por ejemplo, la instalación del servidor Web, que está asociada a la segunda de estas vertientes, nos obligará a hablar del socket 80, del protocolo HTTP, del lenguaje HTML y otras cuestiones que pertenecen a la primera. Posteriormente, la necesidad de ofrecer nuestros servicios Web a través de un servidor proxy, dará paso a temas como la traslación de direcciones IP y la seguridad de nuestra red.

En definitiva, se trata de un repaso por todo el mundo Internet del que sacaremos en claro, una intranet operativa y tangible, y una no tangible pero importante colección de conocimientos y de experiencias.

SOBRE LA CALIDAD DEL SOFTWARE

Uno de los fenómenos más atractivos que se nos presentan en el mundo Internet es el de la existencia de diversas organizaciones que, sin ánimo de lucro, desarrollan y distribuyen software de todo tipo; desde sistemas operativos, compiladores y herramientas de programación, hasta procesadores de textos, hojas de cálculo y aplicaciones de usuario final.

Gran parte de este software es de excelente calidad, superando en multitud de ocasiones a sus homólogos comerciales. No es extraño pues el que muchas empresas decidan incluirlo como parte de su tecnología informática. Habitualmente, cuando una empresa de un cierto nivel decide utilizar FreeBSD o Linux como parte de sus plataformas, no lo hace por regatear las cien mil pesetas (por poner un ejemplo) que viene a costar un Unix comercial, sino porque está convencida de la viabilidad de dichos sistemas. Vamos a detenernos brevemente en dos ejemplos de lo anterior que, además, utilizaremos como parte integrante de nuestra intranet: el sistema operativo Linux y la Licencia Pública GNU.

EL SISTEMA OPERATIVO LINUX

Linux es un clónico del sistema operativo Unix. ¿Qué quiere decir esto? Pues que, desde un punto de vista purista, no es realmente Unix, ya que no se ha utilizado el kernel original de Unix como base, sino que se ha reescrito desde su inicio. Sin embargo, a efectos prácticos, podemos considerar que es Unix, y además un excelente Unix: rápido, robusto y acompañado de un sinfín de software y herramientas. Se distribuye gratuitamente bajo los términos recogidos en la General Public License GNU.

Linux ha sido (y continua siendo) desarrollado por una serie de personas que, compartiendo código, solucionando errores y aportando ideas, mejoran día a día la potencia y versatilidad de este sistema.

Entre las capacidades actuales de Linux, podemos destacar las siguientes:

Incluye los estándares IEEE POSIX.1, System V y BSD, lo que le hacen muy portable y compatible con la mayoría de los Unix del mercado.

Su sistema gráfico está basado en X Windows, el estandar en la mayoría de las implementaciones Unix.

Linux contiene una completa implementación de la familia de protocolos TCP/IP y soporta el rango completo de protocolos, clientes y servicios predominantes en la Internet. Esto le hace, no ya capaz, sino adecuado y aconsejable como plataforma para un sistema que se va a dedicar a proveer o requerir servicios Internet. Matizamos de esta forma algo dicho anteriormente: no vamos a utilizar Linux para ahorrarnos los costes de software en nuestra intranet. Probablemente, en igualdad de condiciones económicas, seguiríamos eligiendo Linux.

Linux soporta la gama completa de productos GNU para desarrollo de aplicaciones, por lo que podemos desarrollar cualquier utilidad que se nos ocurra. Tanto Linux como el software GNU se distribuye con los fuentes incluidos, lo que es una característica inapreciable para el que quiera saber el cómo y porqué de su funcionamiento.

SOBRE LA GNU GENERAL PUBLIC LICENSE

El software acogido a la Licencia Pública GNU se distribuye acompañado de los programas fuente. Permite la modificación de los mismos y su redistribución, así como su uso para crear nuevo software, pero siempre bajo los términos recogidos en la Licencia. La Licencia Pública es la base legal por la que no se ofrece garantía sobre los programas, ya que estos se ofrecen libres de cargos a todo el que los quiera utilizar.

¿Son por tanto programas de dominio público? La respuesta es que no. En un programa de dominio público no se suele distribuir el código y no pertenecen a nadie. Estos programas están bajo copyright, protegidos por derechos de autor, pero licenciados para su redistribución y libre modificación.

Por poner un ejemplo, el kernel de Linux es un programa bajo un copyright (copyright (C) 1993 by Linus Torvalds). Lo mismo ocurre con otras piezas del software de este sistema operativo. Sin embargo se ofrece de manera gratuita bajo los términos de la GPL de GNU.

Para una información completa sobre el tema, recomendamos a nuestros lectores consultar el documento GNU GENERAL PUBLIC LICENSE (Free Software Foundation).

SOBRE EL HARDWARE

A pesar de que gran parte de lo que vamos a hacer se podrían realizar con una sola máquina, (evidentemente, un ordenador puede servirse a sí mismo o, visto desde el punto de vista opuesto, ser cliente de sí mismo), vamos a considerar que tenemos una red Ethernet mínima, es decir, dos ordenadores, cada uno con su tarjeta y el cableado correspondiente. También sería conveniente disponer de un modem y una conexión a la Internet contratada con cualquier proveedor de servicios. Esto nos será muy útil a la hora de obtener software y documentación para nuestra intranet.

Naturalmente, de cuanto mayor número de máquinas dispongamos, tanto mejor. Pero sólo plantearemos la posible necesidad de una tercera máquina dentro de unos meses, cuando nos adentremos en temas tales como el

control del tráfico y la seguridad de nuestras conexiones.

NUESTRO OBJETIVO

El objetivo de esta serie de artículos tiene una doble vertiente: la práctica y la teórica. Evidentemente, a la finalización de la serie habremos obtenido una intranet plenamente configurada y funcional. Pero, no menos importante, (quizá más), tendremos los conocimientos necesarios sobre todo lo que implica su manejo y, por lo tanto, el funcionamiento real de la Internet, su estructura y sus servicios.

No queremos dar una guía de "hágaselo usted mismo" donde escribir "A la pregunta de... Hay que responder que ...". De esta forma, cada uno de los pequeños pasos de configuración de nuestros sistemas servirá de excusa para refrescar, afianzar y, en algunos casos, descubrir el cómo y porqué de los conceptos y tecnología Internet, mediante explicaciones abundantes y, pretendemos, lo más claras que sea posible.

De modo que, con este doble objetivo, comenzamos por el principio.

OBTENCIÓN DEL SISTEMA OPERATIVO

El desarrollo de nuestra intranet se basará en la distribución slackware de Linux, versión 3.1, que es la última versión que se encuentra en la URL ftp.cdrom.com/pub/linux/slackware-3.1 a través de la Internet.

Como ya hemos dicho, Linux es un sistema operativo que se distribuye bajo la GNU General Public License, por lo que puede ser distribuido, copiado y redistribuido de forma libre bajo los términos y condiciones de dicha licencia. De todas formas, cuando tengamos montada la infraestructura de nuestra intranet, suponiendo que algunos de nuestros lectores pueden haber adquirido licencias de otros sistemas operativos comerciales, mostraremos cómo se puede agregar máquinas con esos otros sistemas operativos a nuestra red para hacerlos participar de los servicios Internet.

SOBRE EL DIRECCIONAMIENTO IP

Para comunicar nuestro ordenador con los demás que están conectados a nuestra red ethernet utilizaremos los protocolos de la familia TCP/IP, que son el estándar adoptado para los servicios Internet. Se trata de una gran familia cuyos protocolos de mayor nivel están especializados por servicios, y los de nivel más bajo por actividades y tareas.

Comparado con el modelo de pila ISO-OSI, de siete niveles de protocolos, TCP/IP reduce este número de niveles a cinco. (Recordemos que la pila OSI es un modelo conceptual, y que la implementaciones que lo siguen fielmente, (siempre según opiniones), es porque se las fuerza a seguirlo, no porque haya una necesidad real de hacerlo.

El sistema de direccionamiento IP, mediante el que se determina cuál es el interface de red terminal de una conexión, es de 32 bits y se escribe normalmente mediante cuatro octetos en formato decimal separados por puntos. Se dejan libres el 0, el 255 en cualquier posición, y también el 127 como primer octeto. Hay hardware y software que soportan el uso de estos valores en determinados casos, sin embargo nosotros procuraremos seguir el estándar más riguroso.

Puede haber redes de envergaduras muy distintas, así que se ha definido una estructura para clasificarlas, según su tamaño en tres clases: "A", "B" y "C".

La redes de clase A comienzan por un número entre 1 y 126. Este número es el identificativo de la red, y se usan los otros tres como identificativo del host, en total, 16.387.064 computadoras. Son muy pocas redes y de tamaño gigantesco, como lo era la Arpanet.

Las redes de clase B se usan para grandes organizaciones. Emplean los dos primeros octetos como identificativo de la red, entre el 128.1 y el 191.254. Los dos últimos son para el host, con lo que sale un total de 64.516 computadoras.

Finalmente, las redes de clase C usan tres octetos, en el rango 192.1.1 a 223.254.254, lo que permite 254 hosts por red.

Algunas grandes organizaciones encuentran conveniente el dividir su número de red en "subnets". De esta forma, una red de tipo B, por ejemplo, puede simular internamente varias redes de tipo C. Fuera del entorno de la organización, esa división no tiene efecto.

El valor 0 en una dirección está reservado para máquinas que no conocen su dirección, bien sea la red o subred a la que pertenecen, como su propio número de host.

El valor 255 se usa para "broadcast". Un broadcast es un mensaje que se envía a todos los sistemas de una red.

Muchos sistemas utilizan direcciones que empiezan por 127 para propósitos específicos.

NUESTRA RED

Nuestra red va a ser de tipo C, que es el más adecuado para organizaciones que no plantean requerimientos fuera de lo normal. La dirección que elegiremos para nuestro servidor será la 192.168.1.1 que pertenece a lo que se conoce como "direccionamiento privado".

El direccionamiento privado consiste en la utilización de un rango de direcciones que se reservan en la Internet, para cada una de las clases A, B y C, destinado a su uso interno en las redes conectadas a la misma. Las direcciones que comienzan por 192.168 son privadas dentro de la clase C. Esto garantiza que no haya dos direcciones coincidentes en la Internet, ya que existen mecanismos para hacer que el direccionamiento privado nunca viaje entre redes.

Lo que realmente indica qué ordenadores pertenecen a una red es la "máscara de subred". La que nos corresponde, como red de tipo C, es la 255.255.255.0, que enmascara lo que es la red, es decir, los tres primeros octetos, dejando libre el número de host; el último octeto.

Según esto, los ordenadores de nuestra red utilizarán la dirección 192.168.1.255 como dirección de broadcast.

Como cosa "curiosa", diremos que la red Infovía de Telefónica utiliza la dirección de red 10.0.0.0, (su DNS principal; su "primera máquina", es la 10.0.1.0). Es, por lo tanto, una red privada de tipo A (máscara: 255.0.0.0). Sobre la conveniencia, oportunidad o licitud de ofrecer determinados servicios públicos mediante una red con

direccionamiento privado, no entraremos en más polémicas.

PUNTO DE COMIENZO

Partiremos de la base de que ya tenemos instalado nuestro sistema Linux en el servidor y en una máquina cliente. Instrucciones sobre la instalación de Linux pueden encontrarse en anteriores números de esta revista, en el cdrom de distribución de Linux Slackware o a través de la Internet.

En el próximo número de "Sólo Programadores" comenzaremos a ver todos ficheros de configuración del sistema que es necesario tocar para que ambos ordenadores; cliente y servidor, entren en red, explicando los conceptos básicos que intervienen en la configuración de un sistema con la familia de protocolos TCP/IP.

ANEXO

A continuación incluimos una traducción parcial de la GNU General Public License, que es la que nos permitirá utilizar todo el software "libre" necesario para nuestra intranet. Cualquier software que nosotros podamos escribir y que queramos que sea de libre distribución se puede poner bajo esta licencia, siguiendo las instrucciones que se dan en la misma.

TÉRMINOS Y CONDICIONES

Esta licencia se aplica a cualquier programa u otro trabajo que contenga un aviso puesto por el propietario del copyright diciendo que debe ser distribuido bajo los términos de esta General Public License.

Actividades distintas a la copia, distribución y modificación no están cubiertas por esta Licencia; están fuera de su ámbito. El acto de ejecutar el Programa no está restringido, y la salida producida por el Programa está cubierta sólo si su contenido constituye un trabajo basado en el Programa (independiente haber sido producida por la ejecución del Programa). Lo cuál es así dependiendo de lo que el Programa haga.

Tú puedes copiar y distribuir copias exactas del código fuente del Programa tanto como recibirlas, bajo cualquier medio que suministre publicado visible y apropiadamente el aviso de copyright y la renuncia de garantía; y dar a cualquier otro receptor del Programa una copia de esta Licencia junto con el Programa.

Tú puedes cobrar un precio por el acto físico de transferir una copia, y puedes a tu elección ofrecer una protección de garantía en el intercambio por un precio.

Tú puedes modificar tu copia o copias del Programa o cualquier porción del mismo, formando de este modo un trabajo basado en el Programa y copiar y distribuir tales modificaciones o trabajo bajo los t_rminos de la Sección 1 expuesta arriba, siempre que tœ tambi_n adjuntes todas las siguientes condiciones:

a. Debes generar los ficheros modificados para transmitir aviso resaltado de que tú has cambiado los ficheros, y la fecha de esos cambios.

b. Debes generar cualquier trabajo que tú distribuyas o publiques que, en su total o en parte, contenga o esté derivado del Programa o de alguna parte del mismo, para ser licenciado como un todo y no ser cargado a terceras partes bajo los t_rminos de esta licencia.

c. Si el programa modificado normalmente lee comandos interactivamente mientras está en ejecución, tú debes prepararle para que, cuando empiece a correr para tal uso interactivo en todos los caminos ordinarios, imprima o muestre un anuncio incluyendo un aviso de copyright y de que no hay garantía apropiados (o, sino, diciendo que tœ proporcionas una garantía) y que los usuarios pueden redistribuir el programa bajo esas condiciones, diciendo a los usuarios como pueden ver una copia de esta Licencia. (Excepción: si el Programa por sí mismo es interactivo pero no muestra habitualmente nada del tipo de un anuncio, tu trabajo basado en el Programa no está obligado a mostrar ningœn anuncio).

Estos requerimientos se aplican al trabajo modificado como un todo. Si secciones identificables de este trabajo no se derivan del Programa, y pueden ser razonablemente consideradas como independientes y trabajos separados en sí mismas, entonces, esta Licencias y sus t_rminos no se aplican a tales secciones cuando tœ las distribuyes como trabajos separados. Pero cuando tœ distribuyes las mismas secciones como parte de un todo que es un trabajo basado en el Programa, la distribución del todo debe estar bajo los términos de esta Licencia, cuyos permisos para otros licenciados han de estar extendidas a la totalidad, y de esta forma a cada una y todas las partes indiferentemente de quién las ha escrito.

De este modo, no es la intención de esta sección la renuncia a los derechos o el ataque a tus derechos al trabajo escrito enteramente por ti, más bien, la intención es ejercitar a controlar la distribución de trabajos derivativos o colectivos basados en el Programa.

En adición, la simple agregación de otro trabajo no basado en el Programa al Programa (o a un trabajo basado en el Programa) en un volumen de almacenamiento o en un medio de distribución no pone a aquél bajo el ámbito de esta Licencia.

Debes copiar y distribuir el Programa (o un trabajo basado en _l, bajo la Sección 2) en código objeto o ejecutable bajo los t_rminos de las Secciones 1 y 2 (expuestas arriba) de una de las siguientes maneras:

a. Acompañándolo con el correspondiente código fuente completo legible por la máquina, el cuál debe ser distribuido bajo los términos de las secciones 1 y 2 expuestas arriba sobre un medio acostumbradamente usado para el intercambio de software; o,

b. Acompañándolo con una oferta escrita, válida por al menos tres a–os, para dar a cualquier tercera parte, por un cargo no mayor que tu coste de llevar a cabo la distribución del fuente, una copia completa legible por la máquina del correspondiente código fuente, a ser distribuida bajo los t_rminos de las Secciones 1 y 2 expuestas arriba sobre un medio acostumbradamente usado para el intercambio de software; o,

c. Acompañándolo por la información por ti recibida como oferta para distribuir el correspondiente código fuente. (Esta alternativa está permitida sólo para distribuciones no comerciales y sólo si tœ has recibido el Programa en código objeto o formato ejecutable con tal oferta, de acuerdo con la subsección b expuesta arriba.)

El código fuente de un trabajo es el formato preferido del trabajo para hacer modificaciones sobre él. Para un trabajo ejecutable, el código fuente completo es todo el código fuente para todos los módulos que contiene, más cualquier fichero interface de definición asociado, más los scripts usados para el control de la compilación y la instalación del ejecutable. De cualquier modo, como especial excepción, el código fuente distribuido no necesita incluir nada que sea distribuido habitualmente (tanto en formato fuente como binario) con la mayoría de los componentes (compilador, kernel, etc) del sistema operativo sobre el que corre el ejecutable, a menos que los componentes en sí mismos acompa–en al ejecutable.

Si la distribución del ejecutable o del código objeto se hace ofreciendo acceso a copiarlo de un lugar predeterminado, entonces el ofrecer acceso equivalente a copiar el código fuente del mismo lugar es válido como distribución del código fuente, incluso aunque terceras partes no est_n obligadas a copiar el fuente junto con el código objeto.

No puedes copiar, modificar, sublicenciar o distribuir el Programa excepto como está expresamente designado bajo esta Licencia. Cualquier intento distinto de copiar, modificar, sublicenciar o distribuir el Programa es inválido, y hará finalizar automáticamente tus derechos bajo esta Licencia. De cualquier manera, las partes que han recibido copias o derechos desde ti bajo esta Licencia no verán finalizados sus derechos con tal de que tales partes estén en total conformidad.

Cada vez que redistribuyes el Programa (o cualquier trabajo basado en el Programa), el receptor recibe automáticamente una licencia del licenciador original para copiar, distribuir o modificar el Programa bajo estos términos y condiciones. Tú no puedes imponer ninguna restricción a–adida sobre el ejercicio de los derechos permitidos aquí a los receptores. Tú no eres responsable de forzar la conformidad de terceras partes hacia esta Licencia.

Esta sección tiene el ánimo de clarificar lo que creemos es una consecuencia del resto de esta Licencia.

Cada versión es dada con un número de versión que la distingue. Si el programa especifica un número de versión de esta Licencia que se aplica a ella y a "cualquier versión posterior", tú tienes la opción de seguir los términos y condiciones de esa versión o cualquier versión posterior publicada por la Free Software Foundation. Si el Programa no especifica un número de versión de esta Licencia, tú puedes escoger cualquier versión siempre que haya sido publicada por la Free Software Foundation.

Si deseas incorporar partes del Programa en otros programas libres cuyas condiciones de distribución sean diferentes, escribe al autor para pedirle permiso. Para el software que es copyright de la Free Software Foundation, escribe a la Free Software Foundation; nosotros a veces hacemos excepciones para ello. Nuestra decisión será guiada por los dos objetivos de preservar el status libre de todos los derivados de nuestro software libre y de promocionar la compartición y reutilización del software en general.

Cómo Aplicar Estos Términos en Tus Nuevos Programas

Si tú desarrollas un nuevo programa y quieres que sea de uso público en la mayor medida posible, el mejor camino para conseguir esto es convertirlo en software libre para que cualquiera pueda redistribuir y modificar bajo estos términos.

Para hacer esto, incluye los siguientes avisos con el programa. Lo más seguro es incluirlo al principio de cada fichero fuente para transportar con la mayor efectividad la exclusión de garantía; y cada fichero debe tener al menos la línea "copyright" y un indicador hacia donde se encuentra el aviso completo.

Una línea dando el nombre del programa y una idea de qué es lo que hace.

Copyright (C) 19yy nombre del autor

Este programa es free software; puedes redistribuirlo y/o modificarlo bajo los términos de la GNU General Public License tal como está publicada por la Free Software Foundation; Versión 2 de la Licencia, o (a tu elección) cualquier versión posterior.

Este programa se distribuye con el ánimo de que sea útil, pero SIN NINGUNA GARANTÍA; sin siquiera garantía implicada de COMERCIABILIDAD o IDONEIDAD PARA UN PROPOSITO PARTICULAR. Ver la GNU General Public License para más detalles.

Debes haber recibido una copia de la GNU General Public License unida a este programa; si no es así, escribe a la Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139. También a–ade información de como contactar por correo electrónico y ordinario. Si el programa es interactivo, pon en su salida un corto aviso como este cuando arranque en modo interactivo:

Gnomovision version 69, Copyright (C) 19yy nombre del autor

Gnomovision viene sin ABSOLUTAMENTE NINGUNA GARANTIA; para más detalles escriba íshow wí. Esto es free software, y a usted se le anima a redistribuirlo bajo ciertas condiciones; escriba íshow cí para más detalles.

Los hipotéticos comandos íshow wí y íshow cí deben mostrar las partes apropiadas de la General Public License. Por supuesto, los comandos que tœ uses pueden ser llamados de otra manera en vez de íshow wí y íshow cí; pueden ser incluso clicks de ratón o puntos de menú a elegir de tu programa. Podrías también convenir con la dirección de tu empresa (si trabajas como programador) o de tu centro de enseñanza para firmar una "renuncia de garantía" para el programa si es necesario. Aquí hay un ejemplo, alterando los nombres:

Yoyodyne, Inc., por la presente se renuncia a todos los copyrights relativos al programa íGnomovisioní (con sus pasos de construcción y compilación) escrito por James Hacker.

Firmado en ......, a 1 de Abril de 1997

Esta General Public License no permite la incorporación de tu programa en programas propietarios. Si tu programa es una librería de subrutinas, debes considerar que es sería más útil permitir linkar aplicaciones propietarias con la librería. Si esto es lo que quieres, usa la GNU Library General Public License en vez de esta Licencia.

CORREO DEL LECTOR

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relaccionado con la programación.

Para que sus consultas sean resueltas tienen disponible la dirección de correo electrónico de Sólo Programadores:

solop@towercom.es

P Llevo leyendo Sólo Programadores desde el principio y siempre me ha servido para aclarar muchas dudas que me surgían en el desarrollo de mis estudios.

El problema que les planteo es de criptoaritmética: quiero realizar una suma lógica de letras, consistente en sumar cadenas de letras, de tal forma que dada una suma de dos cadenas y la solución (también en letras), cada letra tomará un valor del 0 al 9. Tendría que conseguir hallar los valores numéricos de cada letra.

Por ejemplo:

```
DIAS
  DE +
 SOL
----
VIVA
```

Me gustaría que me dieran una solución en pseudocódigo o Pascal.

Aprovecho para felicitarles por su estupenda revista.

R La solución se puede conseguir mediante backtracking expandiendo el árbol de búsqueda y explorando el árbol de soluciones en profundidad hasta encontrar la primera válida.

El método de backtraking lo que hace es recorrer todo el árbol de posibles soluciones mientras la solución que se va construyendo sea válida, es decir, cumpla un conjunto de condiciones que son las características propias del problema.

Cuando una solución deja de ser válida se vuelve hasta el punto donde dejó de serlo y esta rama del árbol deja de ser explorada y se continúa buscando las soluciones a partir de ahí.

Para nuestro problema tenemos que considerar ciertos supuestos:

1º.- Los caracteres toman valores entre 0 y 9.

2º.- Si dos caracteres son diferentes sus valores numéricos también han de ser diferentes.

3º.- Como consecuencia de los dos puntos anteriores no pueden existir mas de 10 caracteres distintos.

4º.- La solución no tiene por qué ser única aunque nosotros dejaremos de buscar cuando encontremos la primera.

5º.- Las cadenas comienzan a sumarse por la derecha, es decir, si las cadenas a sumar son LECHE y CAFE la suma se hará de la siguiente forma:

E+E= O;

H+F=D;

C+A=L;

E+C=A; L=C.

LECHE + CAFÉ = CALDO

También nos hace falta determinar el significado de algunos téminos:

1º.- **Conjunto de Caracteres** = Un subconjunto de {A, B, ... , Z} que va a tener como máximo cardinal 10.

2º.- **Conjunto de Valores Candidatos** = Es un conjunto que se va a ir reduciendo a medida que fijemos valores para los caracteres y que es un subconjunto de {0, ..., 9}.

3º.- **Conjunto de Posibles Soluciones** = Son todas las combinaciones que se pueden dar entre la correspondencia que existe entre el Conjunto de Caracteres y el Conjunto de Valores Candidatos, es decir, si por ejemplo el Conjunto de Caracteres es {A, B, C} y el Conjunto de Valores Candidatos es {0, ..., 9} el Conjunto de Posibles Soluciones son todas las combinaciones de tres digitos que se pueden hacer con los valores {0, ..., 9}. Combinaciones sin repetición de 9 elementos tomados de 3 en 3.

4º.- Denotaremos por #(Conjunto) al número de elementos de un conjunto (cardinal de un conjunto).

5º.- La operación factorial de un número se expresará N!, donde N es un número natural>0 cualquiera.

Reducción del Conjunto de Valores Candidatos:

El Conjunto de Posibles Soluciones se puede hacer muy grande dependiendo del cardinal del Conjunto de Valores Candidatos, en general es: #(Conjunto

de Valores Candidatos)! / (#(Conjunto de Valores Candidatos) - #(Conjunto de Caracteres))!, en el caso más completo, Conjunto de Valores Candidatos = {0, ..., 9}, se tendrá que #(Conjunto de Caracteres) = 10 y #(Conjunto de Valores Candidatos) = 10 con lo que #(Conjunto de Posibles Soluciones) = 10! = 3628800 combinaciones, por lo tanto cuanto mas pequeño sea el conjunto de valores candidatos se reduce considerablemente el conjunto de posibles soluciones.

Cuando se fija algún valor candidato el carácter fijado desaparece del Conjunto de Caracteres y el valor que toma este carácter desaparece del Conjunto de Valores Candidatos.

Por lo antes expuesto lo primero que hay que hacer es observar el problema a resolver para ver si se cumplen alguna de las siguientes condiciones que reducen el conjunto de candidatos y por lo tanto el espacio de búsqueda:

1º.- Si la longitud de la cadena solución difiere en una unidad de la longitud de la mayor de las cadenas a sumar, el carácter más a la izquierda de la cadena solución toma el valor 1, con lo que ya se habrá fijado uno de los valores candidatos.

El Conjunto de Caracteres = Conjunto de Caracteres - carácter fijado y el Conjunto de Valores Candidatos se reduce, en el caso mas completo, a {0, 2, 3, 4, 5, 6, 7, 8, 9} con lo que el conjunto de posibles soluciones es 9! = 362880 que es un valor considerablemente inferior al que teníamos antes.

2º.- Si la longitud de la cadena solución difiere en dos unidades de la longitud de la menor de las cadenas a sumar, el carácter más a la izquierda de la cadena solución es 1, el siguiente es 0 y el carácter más a la izquierda de la cadena más larga toma el valor 9. De esta forma habríamos fijado 3 caracteres por lo que #(Conjunto de Posibles Soluciones), en el caso mas completo sería 7! = 5040 .

3º.- Cuando la longitud de las cadenas a sumar sea diferente nos vamos a encontrar con caracteres a los que no se les suma otro carácter, por ejemplo al sumar LECHE + CAFE = CALDO la suma explícita sería:

E+E=O;

H+F=D;

C+A=L;

E+C=A;

y L=C,

donde al carácter L no se le ha sumado ningún otro. En estos casos pueden suceder dos cosas: Voy a plantearlo de forma genérica.

Sean X y Z dos caracteres tales que X pertenece a una de las cadenas a sumar y Z pertenece a la cadena solución, entonces si X es distinto de Z => X=(Z-1) mod 10 y Z=(X+1) mod 10, con lo que una vez que se haya encontrado el valor de uno de los caracteres también se ha encontrado el valor del otro. El caso X=Z no proporciona ninguna información adicional.

Las reglas de la operación Suma de Caracteres del problema propuesto son:

Sean X, Y, Z pertenece al Conjunto de Caracteres y toman valores en {0, ..., 9} y sea una variable arrastre que pertenece a {0, 1}, tales que Z = (X + Y + arrastre) mod 10 . En la primera suma arrastre = 0.

Para las siguientes sumas la variable arrastre vendrá determinada por el valor que haya tomado la expresión X + Y + arrastre en la suma anterior: si X + Y + arrastre <= 9 entonces arrastre = 0 y si X + Y + arrastre > 9 entonces arrastre = 1.

- **Z = (X + Y + arrastre) mod 10,** para todas las sumas excepto la última.
- Al principio arrastre = 0.
- En la última suma Z = X + Y + arrastre, de tal forma que si Z > 9 entonces aparecerán 2 caracteres, uno que corresponde al primer dígito y otro que corresponde al segundo.

Por ejemplo si Z = 10 aparecerán dos caracteres en el siguiente orden primero el que se corresponda con el 1 y luego el que se corresponda con el 0.

CÁLCULO DE LA SOLUCIÓN

Usando el método de backtracking la solución se va construyendo por etapas.

Se dispone de un Conjunto de Valores Candidatos {0, ..., 9} y una correspondencia inyectiva entre el Conjunto de Caracteres y el de Valores Candidatos.

La solución es una tupla de tal forma que si el Conjunto de Caracteres es {C1, ..., Ck} con *k<= 10* la tupla Solución es {N1, ..., Nk} donde Ni <>Nj; para todo i <> j y Ni pertenece al Conjunto de Valores Candidatos.

Para resolver el problema hay que seguir los siguientes pasos:

1º.- Determinar el Conjunto de Caracteres.

2º.- Establecer el Conjunto de Valores Candidatos.

3º.- Establecer una correspondencia inyectiva entre el Conjunto de Caracteres y el Conjunto de Valores Candidatos.

4º.- Fijar el mayor número de valores posibles usando los métodos descritos en el apartado que habla de la reducción del Conjunto de Valores Candidatos.

5º.- A partir de aquí generar el árbol de búsqueda y aplicar la técnica de backtracking para el Conjunto de Valores Candidatos que nos haya quedado.

Un pseudocódigo para la resolución por backtracking sería:

```
Procedure Caracteres (Solucion, i, Salir)

  begin

        Coger el primer valor del Conjunto de Valores Candidatos.

        repeat

        Salir = false;

        Seleccionar el siguiente valor del Conjunto de Valores Candidatos

        if SOLUCION ACEPTABLE then
        begin
                Añadir candidato a la solución:  Solucion[i]= valor;

                if SOLUCIÓN INCOMPLETA then

Caracteres(Solucion, i+1, Salir);

                else (Se encontró la solución)

                        Salir = true;

end
```

- Solucion es una tupla donde se va metiendo los valores solución del problema para construir la solución.

- i es la etapa por la que vamos construyendo la solución.

- Salir se utiliza para terminar el proceso de búsqueda cuando se haya encontrado la primera solución válida.

SOLUCIÓN ACEPTABLE

Se tiene que cumplir que:

- Para todas las sumas de caracteres planteadas a partir de las cadenas a sumar y la solución de la suma y que sean resolubles con los valores calculados por la tupla Solucion se cumplen las reglas de la Operación Suma de Caracteres descritas anteriormente.

- Todos los valores de la tupla Solucion son distintos entre sí.

SOLUCIÓN INCOMPLETA

#(Tupla Solucion) < #(Conjunto de Caracteres).

Julia Sánchez

P *Estoy suscrito a su revista Sólo Programadores desde hace unos meses y no puedo dejar pasar esta ocasión sin felicitarles por la calidad y variedad de los artículos de la misma.*

Mi problema es el siguiente: estoy programando una aplicación de tratamiento de imágenes que ha de ejecutarse sobre plataformas UNIX. Hasta ahora la he ido desarrollando en mi casa donde tengo instalado Windows 95. Lo que me ocurre es que no soy capaz de compilar los programas con arrays mayores de 64k.

Sé que es el límite del tamaño de segmento impuesto por el DOS, pero pensaba que con W95 se podría obviar dicho limite. He utilizado el Visual C++, sin ningún éxito.

¿Realmente no se puede compilar una aplicación para DOS utilizando un modelo de memoria plana ni siquiera con W95?. También tengo la opción del Linux, pero tengo el disco duro a tope y la idea de reparticionarlo no me apetece demasiado. ¿Hay algo que pueda hacer?. Agradeceré cualquier sugerencia que puedan ofrecerme.

Muchas gracias por su atención y mucho ánimo para seguir con el buen trabajo que llevan a cabo.

Luis Ripoll Morales.

R Es muy interesante la pregunta del tratamiento de punteros mayores de 64K en distintos, entornos. Seguro que a partir de ahora no tendrás problemas:

UNIX: en este entorno la utilización de punteros mayores de 64 no tiene ninguna dificultad, y además es totalmente transparente.

```
#include <stdio.h>

#include <malloc.h>

char *puntero;

puntero=(char *)malloc(70000);
....
free(puntero);
```

DOS: aquí si tienes la limitación de 64K pero es fácilmente superada usando los punteros huge.

```
#include <stdio.h>

#include <malloc.h>

char huge *puntero;

/* El primer parámetro es un long e indica el número de elementos

del array y el segundo es el tamaño de cada elemento */

puntero=(char huge *)halloc(70000L,sizeof(char));

.....
hfree(puntero);
```

WINDOWS 16: En Windows tampoco estamos limitados, pero para ello lo más conveniente es utilizar el modelo de memoria largo. Las funciones que debes utilizar son GlobalAlloc, GlobalLock y GlobalFree.

WINDOWS 32: Igual que en Windows 16 sólo que ya no es necesario que compiles en modelo largo, ya que todos los punteros lo son. Las funciones son las mismas:

GlobalAlloc, GlobalLock y GlobalFree.

Francisco E. López Criado

CONTENIDO DEL CD-ROM

ÍNDICE DE REVISTAS, SLACKWARE 96 (LINUX) Y UTILIDADES SHAREWARE

En el CD-ROM correspondiente a este mes de SÓLO PROGRAMADORES, se ha incluido un índice de todas las revistas publicadas hasta la fecha, en formato HTML, que puede ser consultado mediante cualquier browser de red, ya sea el Internet Explorer, cuya versión 3.0 incluimos en el compacto o la vigente de Netscape 3.0, además incluimos una versión del conocido editor Unix Emacs en su versión de dominio público ejecutable en los entornos Windows de 32 bits y las utilidades necesarias para instalarlo, también se incluyen los Plug-Ins necesarios para reproducir los ejemplos de la revista en un navegador de los ya mencionados y, por supuesto, no nos olvidemos de todos los fuentes de los artículos de la revista.

ÍNDICE DE REVISTAS

Localizado en el directorio \Indice del CD-ROM se encuentra el fichero o página de inicio Indice.htm correspondiente al menú principal del mismo.

Para que dispongáis de una forma rápida y cómoda de consultar el contenido de todos los números de Sólo Programadores, hemos creado un índice con todas las revistas. En él, podréis acceder a cada una de las revistas directamente y saber qué temas se han tratado.

Y lo mismo por temas, tenéis los artículos ordenados por temas de interés, para que todos los artículos publicados estén agrupados según la sección en que aparecieron.

Este índice se actualizará todos los meses, incluyéndose mejoras y nuevos hipervínculos para que podáis acceder a información relacionada.

No será necesario copiar el contenido del directorio a nuestro disco duro, aunque podemos hacerlo, ya que está listo para navegar por él. Para poder visualizar dicho índice necesitamos un navegador de Web; podemos encontrar uno de ellos, el de Microsoft en sus correspondientes versiones española e inglesa (Windows 95&NT 4.0 32bits) dentro del directorio \Browsers. Bastará pues con pinchar dos veces en el ejecutable, el cual procederá a autoinstalarse.

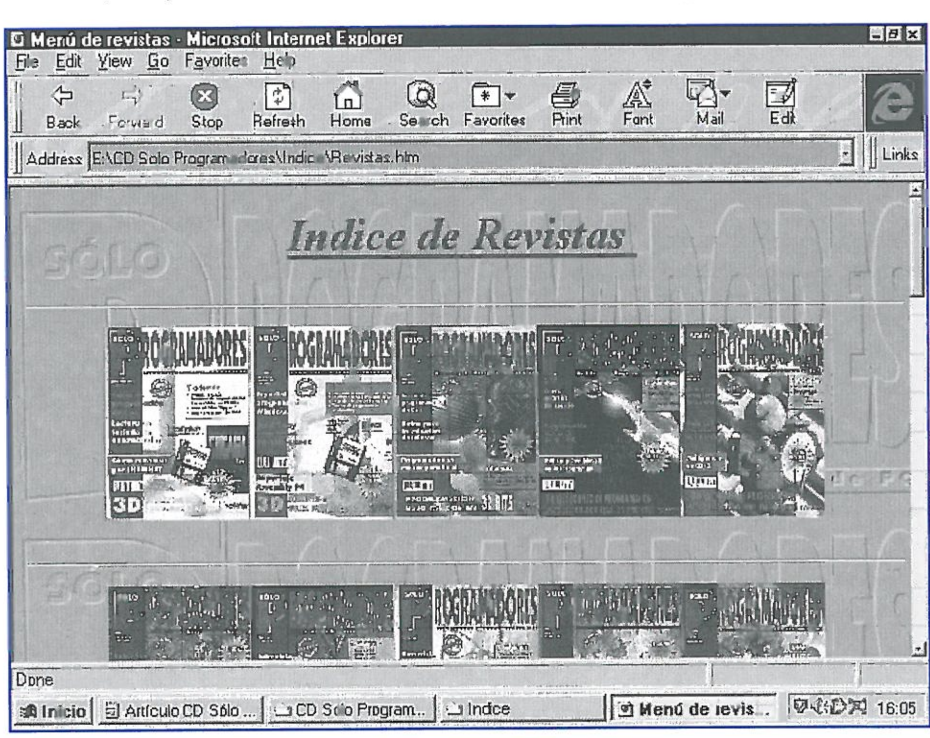
Una vez que dispongamos del browser, tendremos un acceso temático y por revistas de los artículos que nos interesen, en posteriores versiones de este índice se incluirá una busqueda temática de artículos para facilitar el uso de la herramienta por nuestros lectores.

SLACKWARE 96

La última versión de la popular distribución de Linux; para su instalación basta con leer las documentaciones incluidas con ésta o en caso de duda os remitimos a las explicaciones dadas en números anteriores de la revista.

EMACS Y UTILIDADES

En este CD-ROM se incluye la versión para Windows 95/NT del archi-conocido editor de textos Emacs en su versión 19.34. Este editor cuenta con verdaderos fanáticos entre los programadores más veteranos en entornos UNIX; entre sus características más destacables cuenta con un potente lenguaje de



macros que le proporcionan una gran flexibilidad.

Aunque el proceso de instalación es sencillo, se describe aquí brevemente los pasos a seguir (información adicional se encuentra en el fichero "NOTES.TXT"). Copiar los siguientes ficheros a un directorio cuidando de conservar los nombres largos.

EMACS-19.34.*-BIN.TAR.GZ
EMACS-19.34.*-LISP.TAR.GZ

Descomprimir con la utilidad suministrada en el directorio \UTILS del CDROM GZIP.EXE mediante la orden

GZIP -D EMACS-19.34.*-BIN.TAR.GZ
GZIP -D EMACS-19.34.*-LISP.TAR.GZ

Proceder ahora a la reconstrucción del árbol de directorios mediante el programa TAR.EXE incluido también en el directorio \UTILS

TAR -XVF EMACS-19.34.*-
BIN.TAR.GZ
TAR -XVF EMACS-19.34.*-
LISP.TAR.GZ

En ciertos puntos de su ejecución, el programa nos notificará que no ha podido cambiar las fechas de modificación y los permisos de acceso de los ficheros, esto es normal.

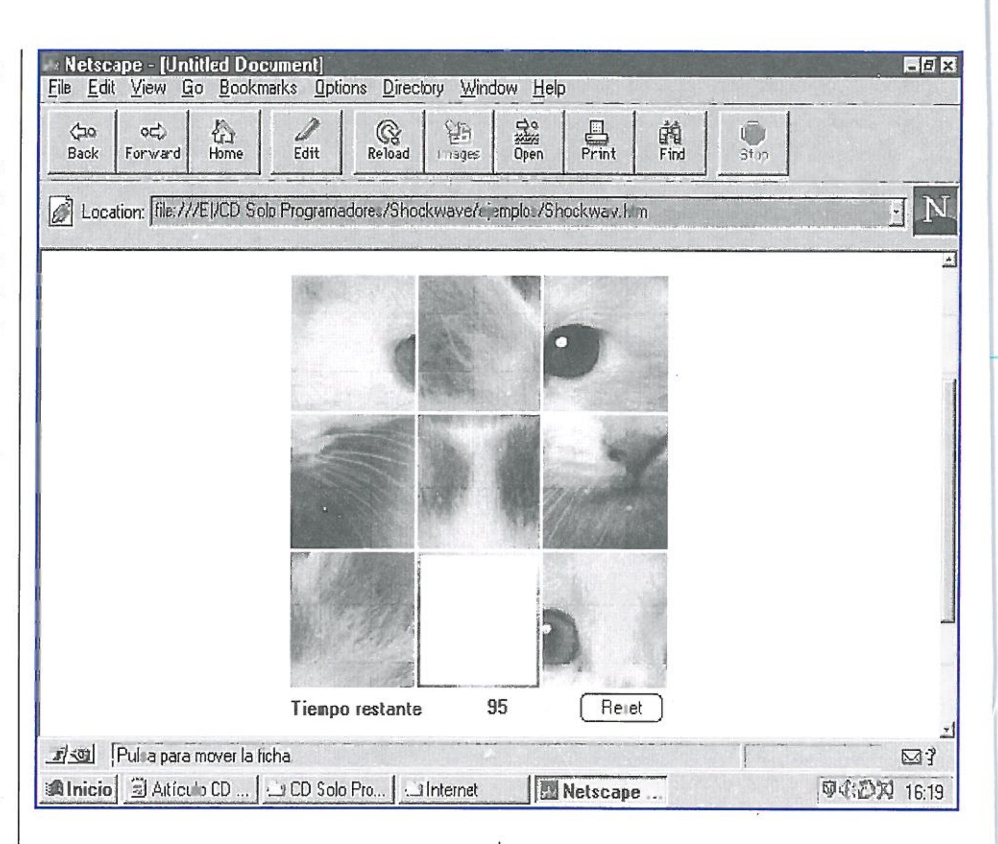
Proceder ahora a la lectura del fichero README incluido en el directorio \emacs-19.34 y seguir sus instrucciones de instalación atentamente.

Para más información o si surgen problemas, se adjunta el FAQ "Frequently Asked Questions" sobre dicho paquete en el presente directorio bajo el nombre "FAQ_Emacs.html" que puede ser visualizado usando cualquier navegador Web.

En este mismo directorio se encuentra también el fichero:

EMACS-19.34.2-UPDATE.ZIP

que es la última actualización existente. Para proceder a su actualización, descomprimir y reconstruir el árbol de directorios de este fichero usando el método anterior y proceder a la sustitución por los nuevos de los ya existentes en el directorio donde se instaló Emacs (aquí \emacs-19.34).



SHOCKWAVE

En el directorio \Shockwave se encuentra el Plug-In correspondiente a los navegadores Microsoft Internet Explorer 3.0 y Netscape Navigator 3.0 (en sus plataformas PC y Macintosh).

Durante la instalación nos dará a elegir en cuál de los dos navegadores queremos instalar dicho Plug-In que nos permitirá trabajar con los ejemplos de la revista.

FUENTES

Por último no nos olvidemos que dentro del directorio \Fuentes nos encontraremos los diferentes códigos fuentes y archivos de ejemplo de la revista.

GUÍA SP

Este número de Sólo Programadores no incluimos la guía SP, pero esperamos continuar con ella en números consecutivos.

Para ello os pedimos que enviéis vuestras páginas HTML con los currículos a la redacción:

Tower Communications, SRL., Aragoneses 7, 28108 Alcobendas, Madrid

con referencia Guía-Sp, para Sólo Programadores.

También podéis enviarlas vía correo electrónico a la dirección solop@towercom.es, indicando en el subject "Guía-SP".

De esta forma, continuaremos intentando desde aquí proporcionar la mejor información sobre los profesionales del sector, y sobre aquellas personas que empiezan, para que desde las empresas puedan contactar fácilmente con vosotros.

SOLICITUD DE PRODUCTOS

Como último punto, nos gustaría que a través de esta dirección de correo electrónico nos hicierais saber aquellos programas, shareware, distribuciones, betas, productos, etc. que os interesan, para incluirlos en el CD-Rom, en la medida en que nos fuera posible. Para eso estamos.